Empirical Asset Pricing via Random Forest

By

Adel W. Malaeb

1132364

Master of Science Quantitative Finance

Institute for Quantitative Business and Economics Research,

Christian Albrechts Universität zu Kiel

Primary Review: Prof. Dr. Alexander Klos

Secondary Review: Prof. Dr. Stefan Reitz

Winter 2020/21

Address: Königsweg 56, 24114 Kiel

Email: stu213387@mail.uni-kiel.de

November 30, 2020

Table of Contents

List of Tables

List of Figures

1 Introduction

In this paper, I implement a Machine Learning model known as Random Forest to investigate the U.S. stock market's expected returns. Machine learning models explain the expected returns' behavior, which is the most widely studied finance problem.

This paper offers two main contributions. First, I use Random Forest to present a descriptive analysis of the best stock-level characteristics that provide an informative value on individual stock returns. Gu et al. (2020) categorize the top 20 most informative predictors into four groups. First, and most importantly, are recent price trend variables such as short-term reversal and industry momentum. Next comes liquidity variables such as the market equity (size) and dollar trading volume. After that are the risk measure variables such as the market beta and idiosyncratic return volatility. Finally, are valuation ratios and fundamental signals such as sales-to-price ratio and the number of recent earnings increase.[1]

Second, I use these best characteristics to predict and measure the individual stock return predictions' accuracy in terms of *out-of-sample R-squared*. I find that in the year 2016, the average R-squared is 17.0743%, which reveals that Random Forest shows considerable promise for asset pricing applications.

This paper will continue in the following way: section 1 provides an overview of previous research in asset pricing literature in addition to a glimpse regarding the importance of Machine Learning on asset pricing. Since it is vital to understand the building blocks of any machine learning model, section 2 discusses the preliminary concepts of Machine Learning models. Considering Random Forest as a Tree-Based Model, a thorough explanation of regression trees and bagged models is discussed before Random Forest, presented in the second part of section 2. Lastly, section 3 presents an empirical study of US equities using Random Forest, which focuses on the

[1] Gu et al. (2020).

main findings for the best stock-level characteristics that explain the behavior of expected returns in the US stock market. Moreover, I will present the prediction of returns for Microsoft and Apple Inc. as a graphical illustration for individual stock return predictions.

1.1 Previous Research

This section will present an overview of various existing literature in finance that justifies the importance of some factors that explain the cross-section of stock returns.

1.1.1 CAPM Beta

The CAPM theory examines the relationship between *risk* and *expected return*, which erects from a portfolio decision.[2]

Hypothesis: A positive association between the average returns and the CAPM's beta exists.[3]

1.1.2 Earnings-to-Price Ratio (E/P)

Basu (1977, p. 663) argues that one should consider the price-to-earnings ratio while assessing a security's future performance. He claims that if the market value is low relative to the earnings (undervalued), it will outperform stocks with high market value. This claim indicates that there exists a relationship between securities' returns and P/E ratios.[4]

Hypothesis: Portfolios that contain low P/E ratio stocks earned average higher returns than portfolios based on high P/E ratio securities after adjusting for risk.[5]

[2] Fama and French (2004), p. 26.
[3] Fama and French (2004), p. 35.
[4] Additional details for empirical testing and overview of the data used are found in Basu (1977, p. 664), in which he investigates his claim using $r_{pt} - r_{ft} = \hat{\delta}_{pf} + \hat{\beta}_{pf}\,(r_{mt} - r_f)$.
[5] Basu (1977), p. 680.

1.1.3 Size / Market Value of Equity (ME)

The size effect has been in existence for many years in different finance literature. Whether the returns are affected by the size or by factors correlated with the size is still unknown. Banz (1981) investigates the relationship between NYSE common stocks' returns and the overall market value.[6]

Hypothesis: Large firms with high market values earn, on average, lower returns than small firms. Therefore, a negative association between excess returns and size exists.[7]

1.1.4 Combination of Size and Book-to-Market Equity BE/ME

Fama and French (1992) argue that if book-to-market equity and size are combined, they will capture the disparity in average stock returns related to β, leverage, and earnings-price ratios. They investigate the mutual roles of size, β, E/P, book-to-market, and leverage for NYSE, NASDAQ, and AMEX average returns.

Beta and Size: The CRSP returns cover the period between 1962 and 1989. When portfolios are built solely on size, a negative correlation between expected return and size appears, which supports the Banz (1981) hypothesis, as stated in section 1.1.3. Also, Fama and French find that β and average returns exhibited a strong positive relation, which supports Sharpe (1964), Lintner (1965), and Black (1972) hypothesis, as stated in section 1.1.1[8]. However, due to a high correlation between size and β, they built size portfolios based on pre-ranked β, which resulted in no association between β and returns.

---

[6] Additional details for empirical testing and overview of the data used are found in Banz (1981), in which he investigates his claim using time-series and cross-sectional regressions defined as: $R_{it} = \gamma_{0t} + \gamma_{1t}\,\beta_{1t} + \gamma_{2t}\left(\frac{\emptyset_{it} - \emptyset_{mt}}{\emptyset_{mt}}\right) + \varepsilon_{it}$.

[7] Banz (1981).

[8] As cited in Fama and French (1992).

BE/ME and Size: Using Fama-MacBeth Regressions, the monthly returns regressed solely on ln (BE/ME) result in an average slope of 0.50 % with a 5.71 t-statistic. On the other hand, using ln (ME) alone results in a t-statistic of 2.58. However, BE/ME does not replace size. When they regress monthly returns on both, the slope of the size remains within a range of -1.99 standard errors from 0; the book-to-market slope remains within a spectacular range of 4.44 standard errors from 0. Therefore, a combination of size and BE/ME is crucial in explaining average returns[9].

BE/ME and Leverage: Fama and French (1992, p. 441) state that BE/ME captures the distress effect, called the *involuntarily leverage effect*, which erects due to the close connection between book-to-market and leverage.

BE/ME, Size, and E/P: Following Ball's (1978) argument[10], current earnings reflect a proxy for future earnings only if firms show positive gains. Thus, Fama and French (1992) represent E/P for negative earnings as a dummy variable. The presence of a book-to-market in the regression eliminates the explanatory capability of the dummy variable. The positive correlation between ln (BE/ME) and E/P explains the relation between average returns and positive E/P[11].

Hypothesis: With β being a weak predictor of average returns, a combination of book-to-market and size explains the variation in average returns of NYSE, AMEX, and NASDAQ. They also capture the leverage and earning-to-price effect on future returns.

[9] Fama and French (1992), p. 441.
[10] As cited in Fama and French (1992), p. 444.
[11] Fama and French (1992), pp. 444-445.

1.1.5 Momentum

Jegadeesh and Titman (1993)[12] state that using a strategy of selling past losers and buying past winners results in approximately 1 % of an excess return per month.

Hypothesis: In general, momentum and excess stock returns exhibit a positive relation.[13]

1.1.6 94-Factors

Cochrane (2011)[14] explains the average US stock returns' behavior by identifying stock level characteristics that contain independent information. Green et al. (2017) take Cochrane's challenge using 94 stock level characteristics to identify significant predictors statistically.

The data set ranges from January 1980 to December 2014, a total of 36 years, with stocks listed on NYSE, NASDAQ, and AMEX. They select one hundred two out of 330 characteristics Green stated in 2013[15]. Within the remaining 102 features, eight had shown large cross-correlation and are therefore excluded, based on the variance inflation factor (VIFs) values[16]. These eight characteristics are dolvol, lgr, momom, betasq, maxiet, quick, stdace, and pchquick.

Since the microcap firms contribute to only 3 % of the NYSE, NASDAQ, and AMEX market cap, most of the analysis is based on non-microcap firms to avoid inferential bias. They implement their study in two ways - *Value*

[12] As cited in Novak (2010), p. 450.
[13] Novak (2010), p. 450.
[14] As cited in Green et al. (2017).
[15] Green et al. (2013).
[16] The variance inflation factor (VIF) is used by Greene (2011) as cited in Green et al. (2017) that shows to which extent a certain characteristic is explained by a linear combination. Green et al. (2017) use a default cutoff value of 7, for which the 8 characteristics discussed above displayed a VIF > 7 and thus were excluded.

weighted least squares (VWLS) and *ordinary least-squares* (OLS) on non-microcap stocks.

They use the Fama MacBeth regression, in which they include 94 characteristics from 1980 to 2014. They mainly focus on the OLS method because its assumptions can significantly identify the most potent independent determinants after accounting for all other factors.

Results:

Six characteristics appear to provide independent information when applying VWLS to all stocks; however, using the OLS to non-microcap firms, the number of significant characteristics increase to nine.

The results are presented in Table 5 in Green et al. (2017, pp. 4409–4410), in which the significant characteristics have a $|t\text{-}statistic| \geq 3.0$. After a combination of the two approaches -VWLS and OLS - twelve features appear to have an essential and significant explanatory power of the average US stock returns: book-to-market, cash, change in the number of analysts, earning announcement return, first-month momentum, sixth-month momentum, numbers of consecutive quarters with earnings higher than during the same quarter in the preceding year, annual R&D to market cap, return volatilely, share turnover, volatilely of share turnover, and zero trading days[17].

1.2 The Importance of Machine Learning on Asset Pricing

Identifying reliable factors that govern the asset pricing phenomena becomes more evident with the use of machine learning. ML algorithms provide hope for better prediction of returns along with practical features of portfolio choice[18]. Gu et al. (2020) apply machine learning models to explain risk premium behavior, the most widely studied finance problem.

[17] A detailed explanation of each characteristic is listed in the Appendix of Green et al. (2017, pp. 4427-4431).
[18] Gu et al. (2020).

1) The anomalies in the existing literature in Asset Pricing models impose a fundamental problem of predicting the expected return across assets. The main specialization of Machine Learning is prediction; it would increase prediction accuracy and identify the essential stock-level characteristics, which govern expected returns' behavior.[19]

2) Green et al. (2013) collect 330 stock-level characteristics, which have a predictive signal. This expansive list of variables is accommodated by Machine Learning models, which are well suited for high-dimensional data[20].

3) Due to highly correlated predictors, traditional methods might break down. Machine learning, on the other hand, condenses variations among predictors by variable selection techniques.[21]

4) Ambiguity of the functional form of predictors imposes a statistical challenge. There exists little evidence of linear interaction among predictors. Thus, diversity in machine learning models approximates complex nonlinear interactions, which results in better prediction measurements.[22]

[19] Gu et al. (2020).
[20] Gu et al. (2020).
[21] Gu et al. (2020).
[22] Gu et al. (2020).

2 Methodology

2.1 Preliminary Concepts of Machine Learning

2.1.1 Definition of Machine Learning

Generally speaking, a machine learns if exposed to an experience and improve future output on similar occasions. These main steps govern the basis of a learning process: Data input, Abstraction, and Generalization. The classic function of a learning algorithm is to represent the raw data in a structured form. In the ensuing abstraction process, a particular model assigns a meaningful representation to the data in which the abstracted knowledge turns into a concrete structure for future actions.[23]In this paper, machine learning describes a statistical predictive model.

2.1.2 Definition of Statistical Learning

Define $X_1, X_2, \dots, X_p$ as a set of input variables or predictors set and let Y be the response or dependent variable. Also, assume that there is a relationship of Y associated with X, defined as:

$$Y = d(x) + \varepsilon \tag{1}$$

The main goal of Statistical Learning is to develop ways to estimate the function $d(x)$ thus to represent the systematic knowledge that X offers about Y.[24]

For prediction, the goal is to predict Y based on an estimate of d using:

$$\hat{Y} = \hat{d}(x) \tag{2}$$

In this case, the exact structure of the function d is not relevant as long as it provides a precise prediction for Y. Therefore, statistical learning methods

[23] Lantz (2013), pp. 10-11.
[24] James et al. (2013), p. 16.

aim to increase predictions' accuracy by minimizing the *reducible error*, defined as the deviation squared from the actual observations.

Assume $\hat{Y} = \hat{d}(x) + \varepsilon$, then

$$E\left[(Y - \hat{Y})\right]^2 = E\left[(d(x) + \varepsilon - \hat{d}(x)\right]^2 = \left[d(x) - \hat{d}(x)\right]^2 + var(\varepsilon)$$

Where $var(\varepsilon)$ is called the *irreducible error* related to the error term ε.

For inference, the goal is to provide a causal relationship between the response variable Y and the predictor X. In other words, to investigate how Y changes when $X_1, X_2, \dots, X_p$ change. [25]

2.1.3 Supervised vs. Unsupervised Learning

According to Lantz (2013), Supervised Learning algorithms construct a predictive model that implicitly forecasts one variable based on other explanatory variables available in the data set. Therefore, given a predictor set $X_i = 1,\dots,n$ an associated dependent variable Y_i is presented.[26] Thus, the learning task is supervised by a variable that provides direct instructions.

On the other hand, Unsupervised Learning is used to develop a descriptive model that does not have a response variable to learn compared to a supervised model. Therefore, predictors have equal importance in the learning process.[27] It is a more complicated process in which the observations in a data set corresponds to explanatory variables $X_i = 1, \dots, n$ with no corresponding response Y_i.[28]

2.1.4 Regression vs. Classification

Two groups of variables exist Quantitative variables, including numerical values and qualitative variables, which include categorical values. Problems

[25] James et al. (2013), pp. 17-19.
[26] James et al. (2013), p. 26.
[27] Lantz (2013), p. 20.
[28] James et al. (2013), p. 26.

with quantitative variables correspond to *regression problems* and those with qualitative variables to *classification problems*.[29]

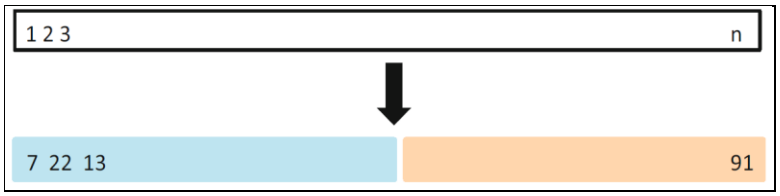
2.1.5 Resampling Methods

When a model fits training samples, it might not project well on new data (test samples); therefore, evaluating the model's performance, known as a *model assessment*, helps obtain additional information about the fitted model. *Resampling methods* frequently draw samples from a training set called the *validation sample*, which will estimate a given model's test error.

Validation Set Approach

The validation set approach includes randomly splitting the available collection of observations into two chunks, a training set, and a validation set. In the training set, the model is fit, and afterward, the validation set predicts the reaction of the observations by estimating the test error rate.[30]

Figure 1. The Validation Set Approach



Note. This figure represents the Validation set approach in which the training set is split into two equal chunks, a training set colored in blue and a validation set colored in yellow - reprinted from "Introduction to Statistical Learning: with Applications in R," by James et al., 2013, p. 17.

To illustrate, consider a set of 306 observations, split into equally weighted chunks of 196 observations each. In this case, the test error estimation using the validation sample might be highly variant since it depends on specific observations in both chunks. Moreover, the size of the training set will shrink, which makes the model inadequate.[31]

[29] James et al. (2013), p. 28.
[30] James et al. (2013), p. 175.
[31] James et al. (2013), p. 177.

Cross-Validation

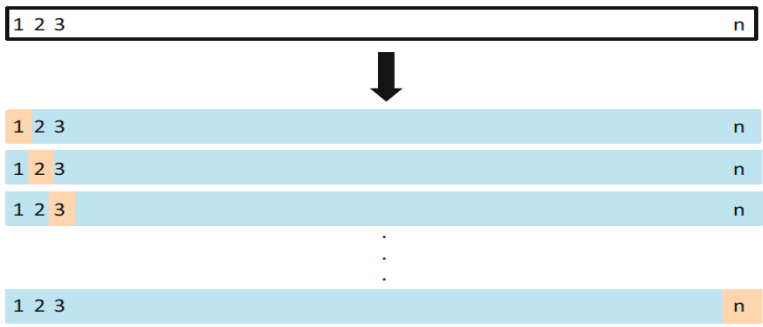
a) Leave-One-Out Cross-Validation

In *leave-one-out cross-validation*, the set also consists of two chunks. The validation set consists of a single observation (x_1, y_1), and the training set contains the remaining $n-1$ observations. The idea is to compute $MSE_1 = (y_1 - \hat{y}_1)^2$ based on the single observation from the validation set, then the procedure is repeated n times to obtain $MSE_1, \dots, MSE_n$.

The cross-validation estimate of the test error is simply the mean of $MSE_1, \dots, MSE_n$ and is defined as:

$$CV_{(n)} = \frac{1}{n} \sum_{i=1}^{n} MSE_i \tag{3}$$

Figure 2. The LOOCV Approach



Note. This figure displays the LOOCV approach where the blue color represents the training set that consists of n data points that include all the observations except for one observation which corresponds to the validation set and shown in yellow- reprinted from "Introduction to Statistical Learning: with Applications in R," by James et al., 2013, p. 179.

One needs to fit the model n times, which makes the leave-one-out cross-validation computationally expensive.[32]

[32] James et al. (2013), p. 179.

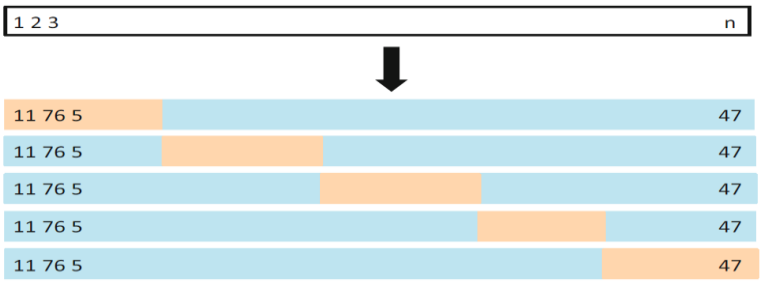
b) K-Fold Cross-Validation

In k-fold cross-validation, the set consists of k folds. The validation set consists of the first fold, and the training set contains the remaining $k-1$ folds. The idea is to compute MSE_1 based on the first fold from the validation set. Then, by repeating the process k-times, we obtain k estimates $MSE_1, MSE_2, \ldots, MSE_k$.

The k-fold cross-validation estimate of the test error is the mean over the k-folds defined as:

$$CV_{(k)} = \frac{1}{k}\sum_{i=1}^{k} MSE_i \tag{4}$$

Figure 3. The K-Fold Cross-Validation



Note. This figure represents a schematic example of k-fold cross-validation in which 5-folds that are colored yellow represent the validation set, and the remaining blue color represents the training set- reprinted from "Introduction to Statistical Learning: with Applications in R, " by James et al., 2013, p. 181.

In this case, there is an evident computational advantage over the leave-one-out cross-validation, which requires fitting the model k-times instead of n-times.[33]

[33] James et al. (2013), p. 181.

2.2 Tree-Based Methods

The nonlinear complexity of individual predictors imposes restrictions on some machine learning models that capture linear interactions only. Gu et al. (2020) investigate the role of a simple linear model estimated by *ordinary least squares* (OLS); however, its empirical failure, in terms of negative out-of-sample R^2, leads to the investigation of other linear models such as *principal components regression* (PCR) and *partial least squares* (PLS). These models, which use dimension reduction techniques, significantly improve predictions by reducing noise and de-correlating highly dependent predictors[34]. To substantially enhance predictions, tree-based models, which capture nonlinearity and allow predictors interactions, are extensively studied in this paper.

2.2.1 Regression Trees

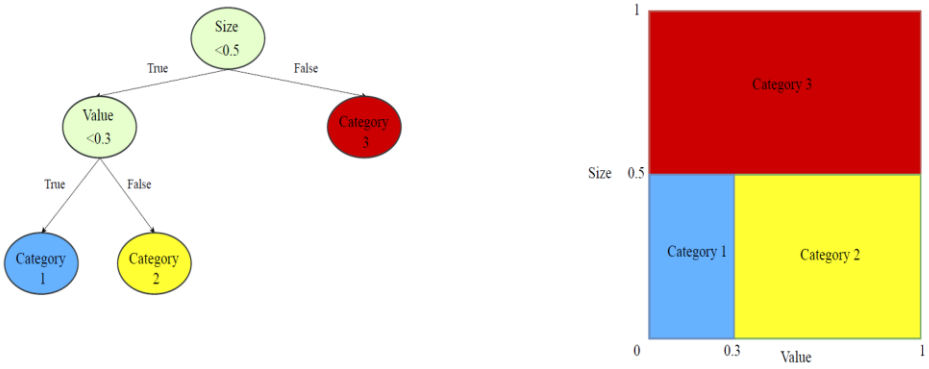
Background

The data set consists of (x, y) where $x \in X$ is a set of predictors space, and $y \in \mathbb{R}$ is the response variable. X is also called the predictors, independent or explanatory variables. Besides, y is also called the response or dependent variable. The predictors' function $d(x)$ takes real values, and ℓ denotes a learning sample or a training set. The data set contains N data points $\{(x_1, y_1), \ldots, (x_N, y_N)\}$ used to predict $d(x)$. Before starting with growing a tree, it is crucial to understand how the predictors' accuracy is measured.[35]

To motivate regression trees, Figure 4[36] illustrates the primary mechanism of this model. In general, tree algorithms try to locate similar observations in which it splits the predictors' space into partitions that contain the response variable.

[34] Gu et al. (2020).
[35] Breiman (1984).
[36] Gu et al. (2020).

Figure 4. Regression Trees



Note. The left diagram represents a regression tree in which the blue, yellow, and red colors correspond to the terminal nodes. The right diagram illustrates a rectangular partitioning based on two characteristics (size and value)- reprinted from "Empirical Asset Pricing via Machine Learning" by Gu et al., 2020.

Consider two predictors: X_1 = "size" and X_2 = "value." The first split at the initial node occurs on "size," observations with less than 0.5 are assigned to a left node, and the others to the right one. The second split occurs on "value," observations that satisfy the condition "value < 0.3" are assigned to the right node. This procedure continues until a terminal node (rectangular partition) is formed at which $d(x)$ is approximated by the average response values at each partition.[37] Before starting with growing a tree, it is crucial to understand how the predictors' accuracy is measured.[38]

A theoretical framework which defines the mean squared error (MSE)

Let N_2 be the size of a test sample $\{(x'_1 y'_1), \dots, (x'_{N_2} y'_N\}$. Given an enormous test sample, an *absolute deviation* measures the accuracy of the prediction defined as:

$$\frac{1}{N_2} \sum_{n=1}^{N_2} |y'_n - d(x'_n)| \tag{5}$$

However, in real practices, a small test sample is used and leads to a classical measure of accuracy known as the *mean squared error*, defined as:

[37] Gu et al. (2020).
[38] Breiman (1984).

$$\frac{1}{N_2}\sum_{n=1}^{N_2}(y'_n - d(x'_n))^2 \tag{6}$$

To define MSE, given that ℓ is fixed, consider $R^*(d) = E(Y - d(x))^2$ as the expectation of the squared error. The basic idea is to minimize $R^*(d)$ using a *Bayes* predictor defined as:

$$d_B(x) = E[Y \mid X = x]^{39} \tag{7}$$

Namely, $d_B(x)$ is the expectation of the response given we observed that $X = x$.

Estimation of the error

The estimation of R^* is done in several ways. Section 2.1.5 discusses some of the methods.

To get the estimate of the test samples error $R^{ts}(d)$, the learning sample ℓ includes two chunks: ℓ_1 & ℓ_2 used to construct the prediction function through ℓ_1, while ℓ_2 is used to form:

$$R^{ts}(d) = \frac{1}{N_2}\sum_{(x_n,y_n)\,\in\,\ell_2}(y_n - d(x_n))^2 \tag{8}$$

However, to estimate the test error based on a validation sample, the *v-fold cross-validation* $R^{cv}_{(d)}$ is used, defined as:[40]

$$R^{cv}_{(d)} = \frac{1}{N}\sum_v \sum_{(x_n,y_n)\,\in\,\ell_v}\left(y_n - d^{(v)}_{(x_n)}\right)^2 \tag{9}$$

Where v corresponds to subsets from the learning sample ℓ each containing the same number of observations. One way to intuitively interpret $R^*(d)$ is to normalize $R^*(d)$ in a way that removes the dependency on the scale.

[39] Section 9.1 in Breiman (1984) provides a proof for equation 7 which is derived according to a simple conditional expectation manipulation.
[40] Section 2.1.5 provides an example of different estimation strategies.

Let $E(y) = \mu$, then $R^*(\mu) = E(Y - \mu)^2$. In this case, a *relative MSE* is used and defined as:

$$RE^*(d) = \frac{R^*(d)}{R^*(\mu)} \tag{10}$$

If there is no information about X, comparing the predictors' performance to that of a baseline predictor μ provides a better understanding of the accuracy. Therefore,

If $RE^*(d) < 1 \Rightarrow R^*(d) < R^*(\mu) \Rightarrow$ the predictors forming $d(x)$ are more accurate than μ.[41]

The process of building a regressions tree

According to Breiman (1984), a regression tree partitions the predictor space X by binary splitting the nodes to reach a terminal node containing a response value $Y(t)$. The procedure of growing a tree is a process of three steps:

1) Developing a method to split the node

2) Determining at which level of the tree the node is terminal

3) A technique to map every terminal node with its corresponding response value $Y(t)$

The last step, in which a response value $Y(t)$ leads to a minimized MSE, is considered the easiest of the three. The value $Y(t)$ defined as:

$$\bar{Y}(t) = \frac{1}{N(t)} \sum_{x_n \in t} y_n \tag{11}$$

Where $N(t)$ is the number of observations that belong to node t. From now on, the notation $R(T)$ will replace $R^*(d)$, such that:

[41] Breiman (1984).

$$R(T) = \frac{1}{N} \sum_{t \in \tilde{T}} \sum_{(x_n \in t)} \left(y_n - \bar{Y}(t)\right)^2 \tag{12}$$

Where T denotes a terminal node.

Choosing the best split at each node

One way to find the best split is to choose a split S^* from a set of splits S, that decreases the magnitude of $R(T)$. To this end, denote by t_{L} & t_{R} as the left and right node respectively and define $\Delta R(s,t) = R(t) - R(t_{\mathrm{L}}) - R(t_{\mathrm{R}})$; therefore, the best split S^* is defined as:

$$\Delta R(s^*,t) = max_{s \,\epsilon\, S}\, \Delta R(s,t) \tag{13}$$

which maximizes a decrease in $R(T)$. Intuitively, this means that the low response values y are separated from the high ones by finding the best split on the independent variables space X.[42]

The following process of growing a tree is known as *recursive binary splitting*. It is also called the *top-down approach* since the splitting always starts at the top of the tree; moreover, a split occurs at a specific step without looking ahead for a better tree in further steps, which results in the name *greedy approach*.[43]

Instability of trees

One can display trees graphically; thus, they are straightforward to explain, which gives them strong interpretability but at the expense of less predictive accuracy. Therefore, they can be non-robust and grow with a high variance - a minor change in data can significantly change the final estimated tree. Due to the hierarchical structure, an error at the first split can grow down with more partitions.[44]Therefore, *bagging* intends to reduce this inherited variance. The next section discusses this method in further detail.

[42] Breiman (1984).
[43] James et al. (2013), p. 306.
[44] Hastie (2009), p. 312.

2.2.2 Bagging or Bootstrap Aggregation

As I mentioned earlier, one disadvantage of growing a single complex tree is that it suffers from high variance, which makes the model unstable. Bagging or bootstrap aggregation, proposed by Breiman (1996), can be used for both regression and classification problems. It is a simple approach that successfully uses bootstrapping to reduce the variance through an aggregation process.[45]

The Bootstrap

The basic idea of bootstrap sampling is to draw samples from a data set, with *replacement*, containing n observations.[46]

Theoretical foundation

Consider the following:

- $X = (x_1, x_2, \dots, x_n)$ a random sample of n observations
- F an unknown probability distribution function
- $\theta = t(F)$ the parameter of interest

The idea now is to estimate θ based on x where $\hat{\theta} = s(x)$, and measure the accuracy of the estimate $\hat{\theta}$. A bootstrap sample, say $x^* = (x_1^*, x_2^*, \dots, x_n^*)$ is drawn randomly from an empirical distribution $\hat{F}$ with a probability to observe a single observation equals to $\frac{1}{n}$. x^* is drawn with replacement, meaning that we might observe that $x_1^* = x_6$, $x_2^* = x_5$, $x_2^* = x_6$, $x_4^* = x_{28}$,..., $x_n^* = x_6$ which intuitively means that a single observation from the initial data set might either appear 0 times or as many times as possible. Given a bootstrap sample x^*, an estimate $\hat{\theta}^* = s(x^*)$ is obtained where $s(x^*)$ has

[45] Kuhn and Johnson (2013), p. 207.
[46] Efron and Tibshirani (1993).

the same properties as $s(x)$ with $\bar{x}^* = \frac{1}{n}\sum_{i=1}^{n} x_i^*$ and $se_{\hat{F}}$ defines the standard error estimate.[47]

Consider the following three steps process:

1- Draw from set X, B independent samples with replacement $x^{*1}, x^{*2}, \dots, x^{*B}$

2- Evaluate $\hat{\theta}^*(b) = s(x^{*b})$ where $b = 1,2, \dots, B$

3- Estimate $se_F\ (\hat{\theta})$ using the B samples by:

$$\widehat{se}_B = \frac{1}{\sqrt{B-1}} \left[\sum_{b=1}^{B} \left(\hat{\theta}^*(b) - \frac{1}{B} \sum_{b=1}^{B} \hat{\theta}^*(b) \right)^2 \right]^{\frac{1}{2}} \qquad (14)$$

Given that $B \to \infty$, $\widehat{se}_B$ converges to $se_{\hat{F}}(\hat{\theta}^*)$: $\lim_{n\to\infty} \widehat{se}_B = se_{\hat{F}} = se_{\hat{F}}(\hat{\theta}^*)$, meaning that the standard deviation of the empirical distribution approaches the population's standard deviation as we draw an infinite number of bootstrap samples.[48]

Bagging

Theoretical foundation

Consider a training sample $\ell = \{(x_n, y_n),\ n = 1, \dots, N\}$ and $d(x, \ell)$ as a function of predictors set. Now, assume there is k training set, ℓ^k, with each k contains N independent data points. Considering the response variable Y as numeric,[49] the procedure would consist of averaging $d(x, \ell)$ over k sets. Therefore, an aggregated prediction function will be given by:

$$d_A(x) = E_\ell\, d(x, \ell) \qquad (15)$$

[47] Figure 6.1 in Efron and Tibshirani (1993, p. 48) provides a graphical representation of the bootstrap algorithm in which it describes the estimation of standard errors in details.
[48] Efron and Tibshirani (1993).
[49] Appendix A.1 provides a proof on why bagging work for numerical predictions.

Where A is a subscript representing aggregation and E_ℓ is the expectation given a training sample ℓ. In this case, one can draw B bootstrap samples from a training sample $\{\ell^B\}$ to form $\{d(x, \ell^B\}$, which yields:

$$d_B(x) = \ avg\ d(x, \ell^B).^{50} \quad (16)$$

Breiman (1994)[51] mentions that the regression tree models are unstable, meaning that a small change in ℓ has a large effect on $d(x)$; therefore, bagging will enhance the accuracy in constructing the prediction function $d(x)$.

Estimation of the test error using out-of-bag observations

Bagged models have a straightforward method to estimate the test error. On average, $\frac{2}{3}$ of the observations appear in each fitted bagged tree and the other $\frac{1}{3}$ observations make out the *out-of-bag* sample. Predicting the response Y for each observation is based on trees grown on OOB observations, which amount to a total of $\frac{B}{3}$ predictions for each observation, and thus the final result is obtained by averaging these predictions.[52] It is somehow identical to the leave-out-one cross-validation method discussed in section 2.1.5.

2.2.3 Random Forest

Definition of Random Forest

Regression trees encapsulate complex interactions between predictors; therefore, they will result in a low bias but high variance model if they are deeply grown. The magnitude of the discrimination that results from bagged trees is equal to that of an individual tree since each tree in the bagged model is identically distributed; thus, the only improvement is achieved through

[50] Breiman (1996).
[51] As cited in Breiman (1996).
[52] James et al. (2013), p. 318.

variance reduction.[53]The Random Forest model that was first proposed by Breiman (2001), which builds an ensemble of randomly grown regression trees, aims to achieve the lowest variance resulting from a single tree or even bagged trees. To illustrate this crucial phenomenon, consider the variance of the aggregated trees over bootstrapped samples defined as: [54]

$$Var\left(\frac{1}{B}\sum_{i=1}^{B} x_i\right) = \rho\sigma^2 + \frac{1-\rho}{B}\sigma^2 \quad (17)$$

Where ρ is the correlation coefficient between predictors $x_i \; and \; x_j$ and B represents the number of bootstrapped samples. Given that the number of bootstrapped sample (B) approaches infinite, the variance reduces to only $\rho\sigma^2$, in which the random forest model will tend to decrease ρ (Correlation between trees in an ensemble) so that a model achieves low bias and variance.[55] The Random Forest's fundamental objective is to improve bagged models by de-correlating trees in the ensemble, which can be done by merely selecting m random subsets from the full set of predictors. In this way, instead of choosing all predictors as a split candidate at each node, only m predictors will be selected at each split. Intuitively, this means that instead of splitting based on a strong predictor only, other predictors will have the chance to be chosen as a split candidate, thus participating in the prediction process.[56]

Variance and de-correlation effect

Given that $B \rightarrow \infty$, it follows from equation 17 that

$$Var\,\hat{d}_{rf}(x) = \rho(x)\,\sigma^2(x) \quad (18)$$

meaning that the variance boils down into two components only $\rho(x)$ & $\sigma^2(x)$.

[53] Hastie et al. (2009), p. 587.
[54] Appendix A.2 provides a detailed proof on how the variance of the aggregated trees boils down to two components only as shown in equation 17.
[55] Hastie et al. (2009).
[56] James et al. (2013), p. 320.

1. $\rho(x) = corr\ [d(x; \theta_1(\ell), d(x; \theta_2(\ell)]$ corresponds to the *sampling correlation* between two aggregated trees.
2. $\sigma^2(x) = Var\ d\big(x; \theta(\ell)\big)$ corresponds to the *sampling variance* of a single tree.

$\rho(x)$ is influenced by random resampling of the learning sample ℓ; in other words, it depends on ℓ due to feature and bootstrap sampling. Also, it is a result of the variability in sampling ℓ. $\rho(x)$ decreases as m decreases, and this is because if different splitting variables are used from different learning sample ℓ, the likelihood of pairs of trees to be similar is low. The total variance of a single tree in an ensemble can be derived using standard conditional variance reasoning defined as:

$$Total\ Variance = \ VAR_\ell\ \hat{d}_{rf}(x) + within\_\ell\_variance. \tag{19}$$

The second component of equation 19 is due to the randomization process; therefore, it increases as m decreases implying that the total variance increases. However, equation 18 shows that the variance of a single tree does not highly depend on the range of m; thus, the ensemble's variance will decrease by more than the increase of a single tree variance. [57]

The theoretical foundation of Random Forests

Denote by $d(x) = E[Y|X = x]$ as the regression function to be estimated using the learning sample $\ell_n = \{(x_1, y_1), \dots, (x_n, y_n)$ with $x_1, \dots, x_n$ & $y_1, \dots, y_n$ being iid random variables. A random forest contains a group of randomized regression trees of the form:

$\{d_n(X, \theta_m, \ell_n), m\ \geq 1\}$ where $\theta_1, \theta_2, \dots, \theta_m$ corresponds to outputs of an iid random variable θ. The estimated aggregated regression trees are of the form:

$$\hat{d}_n(X, \ell_n) = \ E_\theta\ [d_n(X, \theta_m, \ell_n)] \tag{20}$$

[57] Hastie et al. (2009), p. 597.

where E_θ is the expectation for θ conditioned on the learning set ℓ_n and on the predictor space X. Now, consider that $A_n(X;\ \theta)$ is a rectangular cell containing the random partition X of $[0,1]^d$ where $d \geq 2$, then each randomized tree is of the form:

$$d_n(X,\theta) = \frac{\sum_{i=1}^{n} Y_i\ 1[X_i\ \in\ A_n(X;\theta)]}{\sum_{i=1}^{n} 1[X_i\ \in\ A_n(X;\theta)]}\ 1\ E_n(X,\theta) \tag{21}$$

Where,

$$E_n(X,\theta) = \left[\sum_{i=1}^{n} 1\ [X_i\ \in\ A_n(X;\ \theta)\ \neq 0]\right] \tag{22}$$

Meaning that it averages all Y_i for each X_i that belongs in the same rectangular cell. Finally, the estimate of the random forest is: [58]

$$\hat{d}_n = E_\theta[d_n(X,\theta)] = E_\theta\left[\frac{\sum_{i=1}^{n} Y_i\ 1[X_i\ \in\ A_n(X;\theta)]}{\sum_{i=1}^{n} 1[X_i\ \in\ A_n(X;\theta)]}\ 1\ E_n(X,\theta)\right] \tag{23}$$

Biau (2012) mentions that the basic idea is that Breiman's approach is consistent, and its rate of convergence relies on the number of robust features in ℓ_n and not on the number of noise variables.[59]

Hyper-parameter Tuning and Overfitting of Random Forest

Hyper-parameter tuning refers to an optimization procedure for the values that govern the Random Forest model's behavior. These values depend on the data set at hand; therefore, choosing optimal values tends to decrease the observation's likelihood to *overfit* the training data. In the context of machine learning, *overfitting* refers to a phenomenon in which a particular model leads to a set of prediction rules fixed to training data only. Intuitively, this means that the model will most probably perform poorly on new/independent

[58] Biau (2012).

[59] $\hat{d}_n(X)$ is consistent if $E[d_n(x) - d(x)]^2\ \to 0$, as $n\ \to\ \infty$. For detailed explanation on asymptotical analysis refrer to Biau (2012, pp. 1066-1072).

data.[60]Table 1 provides an overview of the hyper-parameters for the Random Forest model.

Table 1. Hyperparameters of Random Forest

Hyperparameter	**Description**	**Typical default values**
mtry	Number of drawn candidate variables in each split	$\sqrt{p}$, $\frac{p}{3}$ for regression
Sample size	Number of observations drawn for each tree	n
Replacement	Draw observations with or without replacement	TRUE (with replacement)
Node size	Terminal node size in terms of minimum observations	1 for classification, 5 for regression
Number of trees	Numbers of trees in the forest	500, 1,000
Splitting rule	Splitting criteria in the nodes	Gini impurity, p value, random

Note. This table provides an overview of the Random Forest model's hyperparameters with a short description and examples of typical default values[61] -reprinted from "Hyperparameters and tuning strategies for the Random Forest," by Probst et al., 2019.

3 Empirical Study of US Equities

Gu et al. (2020) apply machine learning models to explain risk premium behavior, the most widely studied finance problem.

3.1 Data Overview

The data sample covers 60 years, starting in March 1957, ending December 2016. They include a total of 6,200 in each month with individual returns obtained from CRSP for all NYSE, NASDAQ, and AMEX firms. The data set also contains stocks with share codes beyond 10 and 11, prices below $5, and financial firms. One reason for selecting the largest pool of assets is that

[60] Probst et al. (2019).

[61] In this paper two parameters, *mtry* and *number of trees*, are tuned which will be discussed in section 3.5.2.

one can avoid overfitting by increasing the number of observations. The stock-level predictive characteristics consist of 20 monthly, 13 quarterly, and 61 annually updated features that are constructed based on Green et al. (2017) with some minor modifications.[62]They avoid forward-looking bias using monthly updated predictors delayed by at most one month, four months lag for the quarterly ones, and six months for the annual ones. According to Green et al. (2017), 94% of the characteristics contain missing observations (from January 1980 to December 2014); therefore, they replace missing observations with the monthly cross-sectional median for each stock.[63]

3.2 Model Specification

An asset's excess return is:

$$r_{i,t+1} = E_t\left(r_{i,t+1}\right) + \varepsilon_{i,t+1} \tag{24}$$

where $E_t\left(r_{i,t+1}\right) = g^*(Z_{i,t})$

with $i = 1, \dots, N_t$ being the number of stocks presented in the data and $t = 1, \dots, T$ representing months. $Z_{i,t}$ denotes a vector of P-dimensional predictors conditioned on a flexible function $g^*(.)$ with some restrictions. To better predict returns for individual assets, information needs to be leveraged from the entire panel. Therefore $g^*(.)$ is assumed to depend neither on i nor on t. Besides, $g^*(.)$ relies on Z through $Z_{i,t}$, meaning that the model uses information from the i^{th} stock at time t.[64]

[62]Gu et al. (2020) provide a detailed explanation of these 94 characteristics which are listed in table 4, 5, and 6 respectively in Appendix B.1.

[63] Gu et al. (2020).
[64] Gu et al. (2020).

3.3 Sample Splitting and Tuning via Validation

Gu et al. (2020) divide the sample into three disjoint subsamples:

1) Training sample (1957-1974): A total of 18 years, used to fit the random forest model based on tuned parameters.

2) Validation sample (1975-1986): A total of 12 years, used for hyperparameter tuning. This procedure enables searching a set of model parameters that minimize the forecast errors on the validation sample.

3) Test sample (1987-2016): A total of 30 years, used to evaluate the model's performance based on out-of-sample observations.

Gu et al. (2020) chose not to cross-validate, as mentioned in section 2.1.5 but adopted a different approach. Since fitting the model can be computationally expensive, each time they refit the model, they increase the training sample by one year while keeping the validation sample fixed but rolled one year forward.[65]

Tuning the random forest model results in the following hyperparameter: Number of trees: 300, depth: 1-6 and number of features in each split $\epsilon\{3, 5, 10, 20, 30, 50, \dots\}$ [66]

3.4 Main Findings

As mentioned earlier, this paper aims to identify the best stock-level characteristics that explain the expected return exhibited by the stock market. Gu et al. (2020) evaluate the importance of each feature based on the ranking of each of the jth input variables denoted by VI_j. Two approached were used; first, all values of the jth predictor is set to zero while keeping the estimates of

[65] Gu et al. (2020).
[66] Gu et al. (2020).

the model fixed in which a reduction in R^2 is calculated, and second, a metric that provides a summary on how the fitted model is sensitive to a change in each j^{th} predictor is used, which is evaluated by the sum of squared partial derivates (SSD) and defined as:[67]

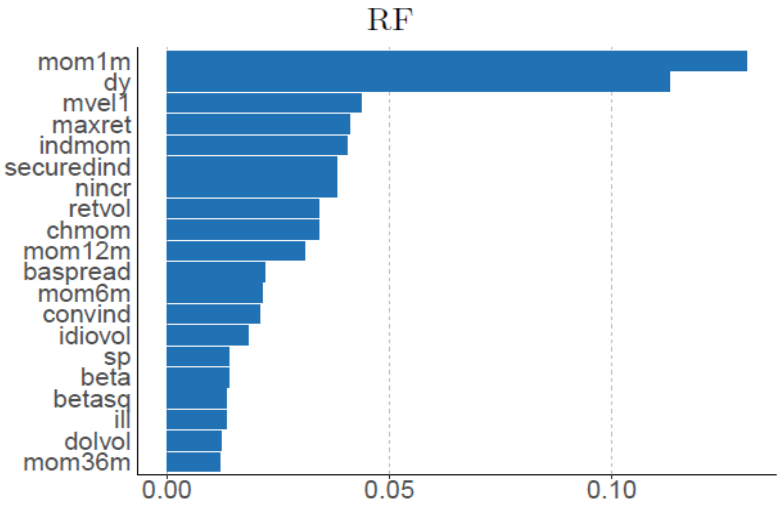
$$SSD_j = \sum_{i,t \in \tau_1} \left(\frac{\partial\, g(z;\theta)}{\partial\, z_j} \mid z = z_{i,t} \right)^2 \tag{25}$$

Figure 10 in Appendix B.2 represents the overall ranking in which the importance of each characteristic is ranked and then summed. Moreover, the top informative predictors are presented on the top, while the least important ones are at the bottom of figure 10.[68]

Figure 5 reports the relative importance of the characteristics modeled by Random Forest only, for which it reveals the top 20 stock-level predictors. Using a random forest model allows a broader set of informative characteristics that explain the expected returns.[69] These characteristics are categorized into four groups:[70]

[67] Dimopoulos et al. (1995) propose the SSD method in the neural networks' literature. However, this method is not applicable for tree-based models. For random forests, Gu et al. (2020) follow the method of mean decrease impurity proposed by Friedman (2001).
[68] 30 recursing training samples were used for average ranking. In addition, figure 10 in Appendix B.2 represents the overall ranking generated by all machine learning models (PLS, PCR, ENet+H, GLM+H, RF, GBRT+H, NN1, NN2, NN3, NN4 and NN5) presented in Gu et al. (2020) study.
[69] Gu et al. (2020).
[70] The SSD methods proposed by Dimopoulos et al. (1995) and the simple deduction in R^2 measure yields nearly the same results for the four groups of characteristics.

Figure 5. Variable Importance By Random Forest



Note. The following figure shows the Random Forest model's most 20 essential characteristics. The measure of importance is normalized to sum to one- reprinted from "Empirical Asset Pricing via Machine Learning," by Gu et al., 2020.

1) Recent price trends: mom1m (short-term reversal), mom12m (stock momentum), maxret (recent maximum return), indmom (industry momentum), mom6m (6-month momentum), chmom (change in 6-month momentum), and mom36m (long-term reversal).

2) Liquidity variables: mvel1 (Size), dolvol (dollar trading volume), ill (illiquidity), and baspread (bid-ask spread).

3) Risk measures: retvol (total return volatility), idiovol (idiosyncratic return volatility), beta (market beta), and betasq (beta-squared).

4) Valuation ratios and fundamental signals: dy (dividend-to-price), securedind (secured debt indicator), convind (convertible debt indicator), sp (sales-to-price), and nincr (number of recent earnings increase).[71]

[71] Gu et al. (2020).

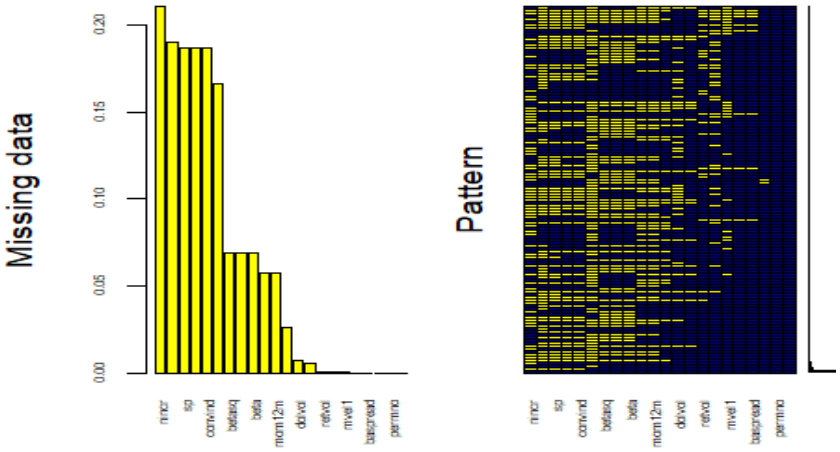
3.5 Return Prediction for Individual Stocks

This section predicts individual stocks' return based on the 20 best stock-level characteristics adopted from section 3.4. The *out-of-sample R-squared* ratio is used to evaluate the Random Forest model's performance on new data. For illustration, two of the largest market capitalization companies in the modified data set, Microsoft and Apple Inc., are used to visualize the predicted returns and compare them with their real/realized returns.

3.5.1 Data Modification

The Data sample covers 12 years, starting in January 2005, ending December 2016, with 144 months.[72] The modified data includes only a total of 2,867 stocks each month. One main reason for this filtration procedure is to have all present firms in each of the 144 months. However, decreasing the number of observations leads to an increase in overfitting the random forest model.

Figure 6. Missing Values



Note. The left figure visualizes the number of missing observations for the sample from January 2005 till December 2016 for each of the 20 stock-level characteristics. Besides, the right one represents the pattern of missing values through time. The algorithm generates this figure follows Appendix C.1.

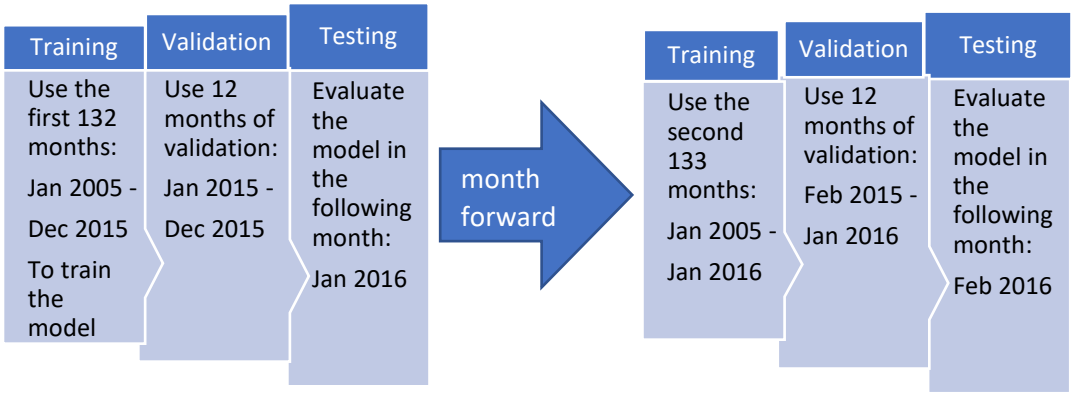
[72] Gu et al. (2020).

Figure 6 shows a high percentage of missing values, which is a severe problem for fitting a random forest model. Therefore, I replace missing observations with the monthly cross-sectional median for each stock.[73]

3.5.2 Sample Splitting, Estimation Procedure, and Hyperparameter Tuning

Considering that the association between stock returns and predictors can vary over time, I estimate the random forest model using a 12-month rolling window.

Figure 7. Sample Splitting and Estimation Procedure



1. Use the first 132 months (Jan 2005-Dec 2015) as the training sample to build the Random Forest model, further divided into training and validation subsamples.
2. Evaluate the model's performance in the following month (Jan 2016).
3. Move forward one month and then use the second 133 months (Jan 2005-Jan 2016) as the training sample, further divided into training and validation subsamples (Feb 2015-Jan 2016). In this way, the training set is increasing one month in size; however, the validation set is kept fixed but rolled one month into the future.
4. Evaluate the model's performance in the following month (Feb 2016).
5. Replicate this procedure until the end of the sample period.

[73] For a full detailed implementation for replacing missing values, refer to Appendix C.1.

Hyperparameter Tuning

As mentioned in section 3.3, Gu et al. (2020) tune three hyper-parameters for the Random Forest model: Depth of trees, number of trees, and number of features in each split. In this section, I adopt $\text{depth} = 1 \sim 6$ for model training; however, for the other two parameters, a random grid search is performed in which table 2 lists the results of the best model.[74]

Table 2. Hyper-Parameter Search Summary

mtries	ntrees	model_ids	residual_deviance
6	400	model_7	0.0041109
6	500	model_12	0.0041137
6	200	model_9	0.0041142
7	200	model_3	0.0041361
9	200	model_8	0.0042071
12	500	model_4	0.0043190
13	300	model_1	0.0043672
14	500	model_6	0.0044131
14	300	model_10	0.0044197
15	200	model_5	0.0044417
15	400	model_2	0.0044445
15	300	model_11	0.0044485

Note. This table summarizes the best model achieved by hyper-parameter random search using H20 described in Appendix C.2. The lowest value of residual deviance represents the best model.

Model 7, a combination of 6 features in each split and 400 trees, leads to minimum residual deviance of 0.0041109.[75]Therefore, in the next section, I use the following parameters to efficiently train a random forest model and predict the US stock market's stock returns.

[74] Random grid search is performed on H2O which is an open source machine learning and predictive analytics platform. Appendix C.2 provides a detailed explanation of random grid search implementation.

[75] Residual deviance is also referred to as MSE (mean squared error), which is a generated metric for supervised machine learning models in H2O.

3.5.3 Model Evaluation

To evaluate the precision of the predictions of the Random Forest model for individual stock returns, I use the *out-of-sample R-squared* defined as:

$$R^2_{oos} = 1 - \frac{\sum_{(i,t) \in \tau_3} \left(r_{i,t+1} - \hat{r}_{i,t+1}\right)^2}{\sum_{(i,t) \in \tau_3} r^2_{i,t+1}} \tag{26}$$

Where τ_3 is the number of fits that corresponds to the testing sample only. One crucial property of equation 26 is that the denominator refers to the sum of squared returns, meaning that predictions are not compared with their respective mean returns, as in many traditional out-of-sample prediction applications. Since historical mean returns are very noisy, using them to predict future returns lowers the *random forest model's goodness of fit*.[76]

Table 3. Monthly Out-of-Sample R-squared

Date	Out_of_sample_R_squared	Model
Jan 2016	0.1702736	RandomForest
Feb 2016	0.2837403	RandomForest
March 2016	0.2729849	RandomForest
April 2016	0.1780589	RandomForest
May 2016	0.0780084	RandomForest
June 2016	0.1320760	RandomForest
July 2016	0.0518337	RandomForest
August 2016	0.1368939	RandomForest
Sep 2016	0.2909351	RandomForest
Oct 2016	0.1909491	RandomForest
Nov 2016	0.1963398	RandomForest
Dec 2016	0.0668220	RandomForest

Note. This table reports the monthly out-of-sample R-squared generated by the Random Forest model following the algorithm presented in Appendix C.3.

R^2_{oos} ranges between a minimum value of 5.18337% to a maximum of 29.09351%, with a mean value of R^2_{oos} over the year 2016 of 17.0743%. Therefore, on average, using Random Forest, 17.0743% of the variation in the expected returns in the year 2016 of the individual stocks is explained by the 20 most informative stock-level characteristics presented in section 3.4.

[76] Gu et al. (2020).

Gu et al. (2020) reports an R^2_{oos} of 0.33% using Random Forest on all 6,200 stocks with 94 stock-level characteristics, as mentioned in section 3.1. This difference in R^2_{oos} results from using only 20 best predictors instead of 94, so in this case, the Random Forest model predicts stock returns based on signal rather than noise. Therefore, this increase in R^2_{oos} also supports the conclusion regarding the top informative predictors in section 3.4.

3.5.4 Stocks Return Predictions

Figure 8. Predicted vs. True Returns of Microsoft in 2016

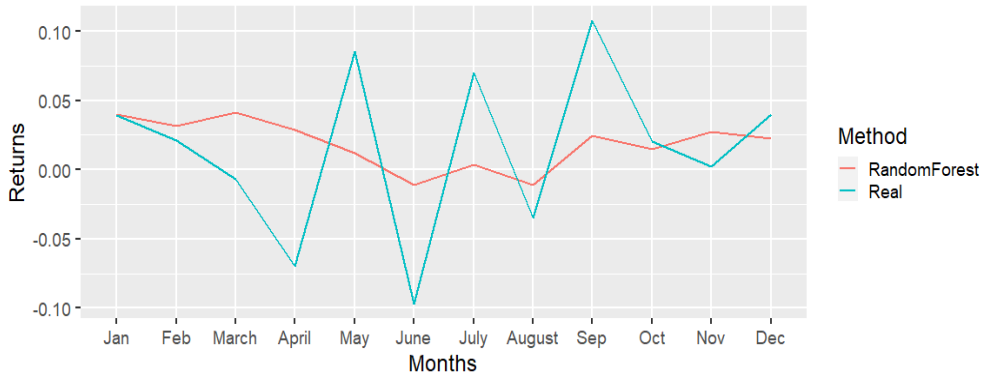
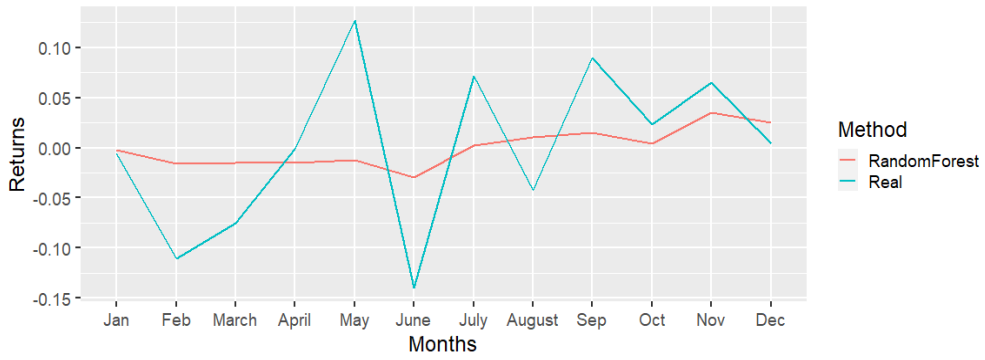


Figure 9. Predicted vs. True Returns of Apple Inc. In 2016



4 Conclusion

This paper identifies the best 20 stock-level characteristics out of 94 features examined by Gu et al. (2020) that have explanatory power in explaining the excess returns exhibited by the US stock market. It also aims to implement the Random Forest model that incorporates these characteristics to predict the individual's stock returns.

Asset pricing applications bring challenges due to a low signal-to-noise ratio. Machine Learning algorithms, especially Random Forest, can predict returns based on the signal; thus, leading to better measurement. The Success of Random Forest based on a significant *out-of-sample R-squared* ratio brings hope for optimal portfolio decisions.

References

Banz, R.W. (1981). The relationship between return and market value of common stocks. *Journal of Financial Economics*, *9*(1), 3-18. www.doi.org/10.1016/0304-405x(81)90018-0

Basu, S. (1977). Investment Performance of Common Stocks in Relation to Their Price-Earnings Ratios: A Test of the Efficient Market Hypothesis. *The Journal of Finance*, *32*(3), 663-682. www.jstor.org/stable/2326304

Biau, G. (2012). Analysis of a random forests model. *The Journal of Machine Learning Research*, *13*(1), 1063-1095. https://dl.acm.org/doi/abs/10.5555/2503308.2343682#d1477194e1

Breiman, L. (1996). Bagging predictors. *Machine Learning, 24,* 123-140. https://doi.org/10.1007/BF00058655

Breiman, L. (2001). Random Forests. *Machine Learning, 45,* 5-32. https://doi.org/10.1023/A:1010933404324

Breiman, L., Friedman, J., Olshen, R.A., & Stone, C.J. (1984). *Classification and Regression Trees.* Chapman and Hall/CRC.

Dimopoulos, Y., Bourret, P., & Lek, S. (1995). Use of some sensitivity criteria for choosing networks with good generalization ability. *Neural Processing Letters*, *2*(6), 1-4. www.doi.org/10.1007/BF02309007

Efron, B., & Tibshirani, R.J. (1993). *An Introduction to the Bootstrap.* Chapman and Hall/CRC.

Fama, E.F., & French, K.R. (1992). The Cross-Section of Expected Stock Returns. *The Journal of Finance*, *47*(2), 427-465. https://doi.org/10.1111/j.1540-6261.1992.tb04398.x

Fama, E.F., & French, K.R. (2004). The Capital Asset Pricing Model: Theory and Evidence. *Journal of Economic Perspectives*, *18*(3), 25-46. www.doi.org/10.1257/0895330042162430

Friedman, J.H. (2001). Greedy Function Approximation: A Gradient Boosting Machine. *The Annals of Statistics*, *29*(5), 1189-1232. www.jstor.org/stable/2699986

Green, J., Hand J.R.M., & Zhang, X.F. (2017). The Characteristics that Provide Independent Information about Average U.S. Monthly Stock Returns. *The Review of Financial Studies*, *30*(12), 4389-4436. www.doi.org/10.1093/rfs/hhx019

Green, J., Hand, J.R.M., & Zhang, X.F. (2013). The supraview of return predictive signals. *Review of Accounting Studies, 18,* 692-730. https://doi.org/10.1007/s11142-013-9231-1

Gu, S., Kelly, B., & Xiu, D. (2020). Empirical Asset Pricing via Machine Learning. *The Review of Financial Studies*, *33*(5), 2223-2273. www.doi.org/10.1093/rfs/hhaa009

Gu, S., Kelly, B., & Xiu, D. (2019). *Autoencoder Asset Pricing Models*. Chicago Booth Research Paper No. 19-24. Available at: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3335536

Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The Elements of Statistical Learning*. Springer. https://link.springer.com/book/10.1007/978-0-387-84858-7

Jegadeesh, N., & Titman, S. (1993). Returns to Buying Winners and Selling losers: Implications for Stock Market Efficiency. *The Journal of Finance*, *48*(1), 65-91. www.jstor.org/stable/2328882

James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). *An Introduction to Statistical Learning: With Applications in R*. Springer. https://link.springer.com/book/10.1007/978-1-4614-7138-7

Kuhn, M., & Johnson, K. (2013). *Applied Predictive Modeling.* Springer.

Lantz, B. (2013). *Machine Learning with R.* Packt Publishing.

Novak, J. (2010). CAPM beta, Size, Book-to-Market, and Momentum in Realized Stock Returns. *Czech Journal of Economics and Finance (Finance a uver)*, *60*(5), 447-460. www.researchgate.net/publication/226649848

Probst, P., Wright, M.N., & Boulesteix, A-L. (2019). Hyperparameters and tuning strategies for random forest. *WIREs Data Mining and Knowledge Discovery*, *9*(3). https://doi.org/10.1002/widm.1301

Appendix A

A.1 Proof for Bagging of Numerical Predictions

Denote by P as the probability distribution function in which each case (y, x) in ℓ is drawn independently. y is assumed to be a numerical value, and $d(x, \ell)$ is the prediction function in which it is aggregated over ℓ yielding the following averaged prediction:

$$d_A(x) = E_\ell \, d(x, \ell) \tag{27}$$

Given that x is fixed,

$$E_\ell \, (y - d(x, \ell))^2 = y^2 - 2y \, E_\ell \, d(x, \ell) + E_\ell \, d^2(x, \ell) \tag{28}$$

Now the idea is to replace $E_\ell \, d(x, \ell)$ by $d_A(x)$, and apply the inequality to the third term of equation 28,

$E_\ell \, d^2(x, \ell) \geq [E_\ell \, d(x, \ell)]^2$ to get:

$$E_\ell \, (y - d(x, \ell))^2 \geq (y - d_A(x))^2 \tag{29}$$

Finally, integrating both sides of equation 29 with respect to x and y, results is an MSE for $d_A(x)$ which is smaller than the MSE of $d(x, \ell)$ averaged over ℓ.

A.2 Proof of the Variance Reduction Equation 17

Assume that $x_i \sim N(m, \sigma^2)$ and denote by ρ as the correlation coefficient between predictors x_i and x_j.

Recall that:

$Var(x) = E[X^2] - [E(X)]^2$ and

$$\rho = \frac{1}{\sigma^2} E\left[(x_i - m)(x_j - m)\right] > 0$$

$$\rho = \frac{1}{\sigma^2} E\left[x_i x_j - m x_i - m x_j + m^2\right]$$

$$\rho = \frac{1}{\sigma^2} \left[E(x_i x_j) - m E(x_i) - m E(x_j) + m^2\right]$$

$$\rho = \frac{E(x_i x_j) - m^2}{\sigma^2}$$

$$\Rightarrow E(x_i x_j) - m^2 = \rho\sigma^2$$

$$\Rightarrow E(x_i x_j) = \rho\sigma^2 + m^2$$

Therefore,

- $E[x_i] = m$
- $E(x_i x_j) = \begin{cases} \rho\sigma^2 + m^2 & if\ i \neq j \\ \sigma^2 + m^2 & if\ i = j \end{cases}$

The variance of the mean over B bootstrapped samples is given by:

$$Var\left(\frac{1}{B}\sum_{i=1}^{B} x_i\right) = \frac{1}{B^2} Var\left(\sum_{i=1}^{B} x_i\right) = \frac{1}{B^2}\left[E\left[\left(\sum_{i=1}^{B} x_i\right)^2\right] - E\left[\sum_{i=1}^{B} x_i\right]^2\right]$$

$$E\left[\sum_{i=1}^{B} x_i\right] = \sum_{i=1}^{B} E(x_i) = Bm$$

$$E\left[\sum_{i=1}^{B} x_i\right]^2 = \sum_{i,j}^{B} E\left(x_i x_j\right) = B\,E[x_i^2] + (B^2 - B)\,E\left[x_i x_j\right]$$

$$= B\,(\sigma^2 + m^2) + B(B-1)(\rho\sigma^2 + m^2)$$

$$= B\sigma^2 + Bm^2 + B^2\rho\sigma^2 + B^2m^2 - B\rho\sigma^2 - Bm^2$$

$$\Rightarrow Var\left(\frac{1}{B}\sum_{i=1}^{B} x_i\right) =$$

$$= \frac{1}{B^2}\,[B\sigma^2 + B^2\rho\sigma^2 + B^2m^2$$

$$- B\rho\sigma^2 - B^2m^2]$$

$$= \frac{B\sigma^2}{B^2} + \frac{B^2\rho\sigma^2}{B^2} - \frac{B\rho\sigma^2}{B^2}$$

$$= \rho\sigma^2 + \frac{(1-\rho)\,\sigma^2}{B}$$

Appendix B

B.1 Details of the 94 Stock-Level Characteristics

Table 4. Details of the Characteristics from 1 to 31

No.	Acronym	Firm characteristic	Paper's author(s)	Year, Journal	Data Source	Frequency
1	absacc	Absolute accruals	Bandyopadhyay, Huang & Wirjanto	2010, WP	Compustat	Annual
2	acc	Working capital accruals	Sloan	1996, TAR	Compustat	Annual
3	aeavol	Abnormal earnings announcement volume	Lerman, Livnat & Mendenhall	2007, WP	Compustat+CRSP	Quarterly
4	age	# years since first Compustat coverage	Jiang, Lee & Zhang	2005, RAS	Compustat	Annual
5	agr	Asset growth	Cooper, Gulen & Schill	2008, JF	Compustat	Annual
6	baspread	Bid-ask spread	Amihud & Mendelson	1989, JF	CRSP	Monthly
7	beta	Beta	Fama & MacBeth	1973, JPE	CRSP	Monthly
8	betasq	Beta squared	Fama & MacBeth	1973, JPE	CRSP	Monthly
9	bm	Book-to-market	Rosenberg, Reid & Lanstein	1985, JPM	Compustat+CRSP	Annual
10	bm_ia	Industry-adjusted book to market	Asness, Porter & Stevens	2000, WP	Compustat+CRSP	Annual
11	cash	Cash holdings	Palazzo	2012, JFE	Compustat	Quarterly
12	cashdebt	Cash flow to debt	Ou & Penman	1989, JAE	Compustat	Annual
13	cashpr	Cash productivity	Chandrashekar & Rao	2009, WP	Compustat	Annual
14	cfp	Cash flow to price ratio	Desai, Rajgopal & Venkatachalam	2004, TAR	Compustat	Annual
15	cfp_ia	Industry-adjusted cash flow to price ratio	Asness, Porter & Stevens	2000, WP	Compustat	Annual
16	chatoia	Industry-adjusted change in asset turnover	Soliman	2008, TAR	Compustat	Annual
17	chcsho	Change in shares outstanding	Pontiff & Woodgate	2008, JF	Compustat	Annual
18	chempia	Industry-adjusted change in employees	Asness, Porter & Stevens	1994, WP	Compustat	Annual
19	chinv	Change in inventory	Thomas & Zhang	2002, RAS	Compustat	Annual
20	chmom	Change in 6-month momentum	Gettleman & Marks	2006, WP	CRSP	Monthly
21	chpmia	Industry-adjusted change in profit margin	Soliman	2008, TAR	Compustat	Annual
22	chtx	Change in tax expense	Thomas & Zhang	2011, JAR	Compustat	Quarterly
23	cinvest	Corporate investment	Titman, Wei & Xie	2004, JFQA	Compustat	Quarterly
24	convind	Convertible debt indicator	Valta	2016, JFQA	Compustat	Annual
25	currat	Current ratio	Ou & Penman	1989, JAE	Compustat	Annual
26	depr	Depreciation / PP&E	Holthausen & Larcker	1992, JAE	Compustat	Annual
27	divi	Dividend initiation	Michaely, Thaler & Womack	1995, JF	Compustat	Annual
28	divo	Dividend omission	Michaely, Thaler & Womack	1995, JF	Compustat	Annual
29	dolvol	Dollar trading volume	Chordia, Subrahmanyam & Anshuman	2001, JFE	CRSP	Monthly
30	dy	Dividend to price	Litzenberger & Ramaswamy	1982, JF	Compustat	Annual
31	ear	Earnings announcement return	Kishore, Brandt, Santa-Clara & Venkatachalam	2008, WP	Compustat+CRSP	Quarterly

Note. Tables 4, 5, and 6 lists the details of all the 94 stock-level characteristics used in the empirical study. Green et al. (2017) initially constructed these features- reprinted from "Empirical Asset Pricing via Machine Learning" by Gu et al., 2020.

Table 5. Details of the Characteristics from 32 to 62 (Continued)

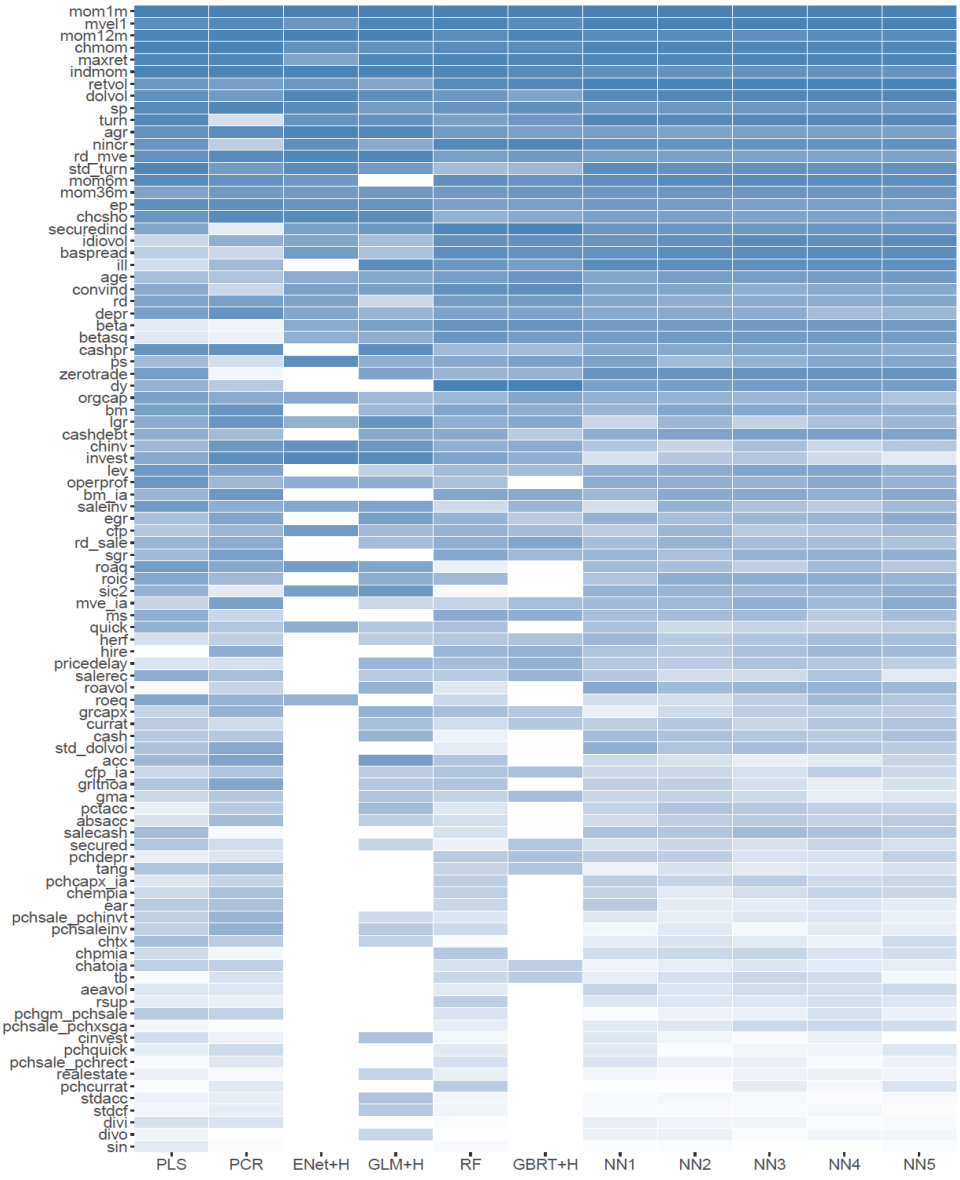
No.	Acronym	Firm characteristic	Paper's author(s)	Year, Journal	Data Source	Frequency
32	egr	Growth in common shareholder equity	Richardson, Sloan, Soliman & Tuna	2005, JAE	Compustat	Annual
33	ep	Earnings to price	Basu	1977, JF	Compustat	Annual
34	gma	Gross profitability	Novy-Marx	2013, JFE	Compustat	Annual
35	grCAPX	Growth in capital expenditures	Anderson & Garcia-Feijoo	2006, JF	Compustat	Annual
36	grltnoa	Growth in long term net operating assets	Fairfield, Whisenant & Yohn	2003, TAR	Compustat	Annual
37	herf	Industry sales concentration	Hou & Robinson	2006, JF	Compustat	Annual
38	hire	Employee growth rate	Bazdresch, Belo & Lin	2014, JPE	Compustat	Annual
39	idiovol	Idiosyncratic return volatility	Ali, Hwang & Trombley	2003, JFE	CRSP	Monthly
40	ill	Illiquidity	Amihud	2002, JFM	CRSP	Monthly
41	indmom	Industry momentum	Moskowitz & Grinblatt	1999, JF	CRSP	Monthly
42	invest	Capital expenditures and inventory	Chen & Zhang	2010, JF	Compustat	Annual
43	lev	Leverage	Bhandari	1988, JF	Compustat	Annual
44	lgr	Growth in long-term debt	Richardson, Sloan, Soliman & Tuna	2005, JAE	Compustat	Annual
45	maxret	Maximum daily return	Bali, Cakici & Whitelaw	2011, JFE	CRSP	Monthly
46	mom12m	12-month momentum	Jegadeesh	1990, JF	CRSP	Monthly
47	mom1m	1-month momentum	Jegadeesh & Titman	1993, JF	CRSP	Monthly
48	mom36m	36-month momentum	Jegadeesh & Titman	1993, JF	CRSP	Monthly
49	mom6m	6-month momentum	Jegadeesh & Titman	1993, JF	CRSP	Monthly
50	ms	Financial statement score	Mohanram	2005, RAS	Compustat	Quarterly
51	mvel1	Size	Banz	1981, JFE	CRSP	Monthly
52	mve_ia	Industry-adjusted size	Asness, Porter & Stevens	2000, WP	Compustat	Annual
53	nincr	Number of earnings increases	Barth, Elliott & Finn	1999, JAR	Compustat	Quarterly
54	operprof	Operating profitability	Fama & French	2015, JFE	Compustat	Annual
55	orgcap	Organizational capital	Eisfeldt & Papanikolaou	2013, JF	Compustat	Annual
56	pchcapx_ia	Industry adjusted % change in capital expenditures	Abarbanell & Bushee	1998, TAR	Compustat	Annual
57	pchcurrat	% change in current ratio	Ou & Penman	1989, JAE	Compustat	Annual
58	pchdepr	% change in depreciation	Holthausen & Larcker	1992, JAE	Compustat	Annual
59	pchgm_pchsale	% change in gross margin - % change in sales	Abarbanell & Bushee	1998, TAR	Compustat	Annual
60	pchquick	% change in quick ratio	Ou & Penman	1989, JAE	Compustat	Annual
61	pchsale_pchinvt	% change in sales - % change in inventory	Abarbanell & Bushee	1998, TAR	Compustat	Annual
62	pchsale_pchrect	% change in sales - % change in A/R	Abarbanell & Bushee	1998, TAR	Compustat	Annual

Table 6. Details of the Characteristics from 63 to 94 (Continued)

No.	Acronym	Firm characteristic	Paper's author(s)	Year, Journal	Data Source	Frequency
63	pchsale_pchxsga	% change in sales - % change in SG&A	Abarbanell & Bushee	1998, TAR	Compustat	Annual
64	pchsaleinv	% change sales-to-inventory	Ou & Penman	1989, JAE	Compustat	Annual
65	pctacc	Percent accruals	Hafzalla, Lundholm & Van Winkle	2011, TAR	Compustat	Annual
66	pricedelay	Price delay	Hou & Moskowitz	2005, RFS	CRSP	Monthly
67	ps	Financial statements score	Piotroski	2000, JAR	Compustat	Annual
68	quick	Quick ratio	Ou & Penman	1989, JAE	Compustat	Annual
69	rd	R&D increase	Eberhart, Maxwell & Siddique	2004, JF	Compustat	Annual
70	rd_mve	R&D to market capitalization	Guo, Lev & Shi	2006, JBFA	Compustat	Annual
71	rd_sale	R&D to sales	Guo, Lev & Shi	2006, JBFA	Compustat	Annual
72	realestate	Real estate holdings	Tuzel	2010, RFS	Compustat	Annual
73	retvol	Return volatility	Ang, Hodrick, Xing & Zhang	2006, JF	CRSP	Monthly
74	roaq	Return on assets	Balakrishnan, Bartov & Faurel	2010, JAE	Compustat	Quarterly
75	roavol	Earnings volatility	Francis, LaFond, Olsson & Schipper	2004, TAR	Compustat	Quarterly
76	roeq	Return on equity	Hou, Xue & Zhang	2015, RFS	Compustat	Quarterly
77	roic	Return on invested capital	Brown & Rowe	2007, WP	Compustat	Annual
78	rsup	Revenue surprise	Kama	2009, JBFA	Compustat	Quarterly
79	salecash	Sales to cash	Ou & Penman	1989, JAE	Compustat	Annual
80	saleinv	Sales to inventory	Ou & Penman	1989, JAE	Compustat	Annual
81	salerec	Sales to receivables	Ou & Penman	1989, JAE	Compustat	Annual
82	secured	Secured debt	Valta	2016, JFQA	Compustat	Annual
83	securedind	Secured debt indicator	Valta	2016, JFQA	Compustat	Annual
84	sgr	Sales growth	Lakonishok, Shleifer & Vishny	1994, JF	Compustat	Annual
85	sin	Sin stocks	Hong & Kacperczyk	2009, JFE	Compustat	Annual
86	sp	Sales to price	Barbee, Mukherji, & Raines	1996, FAJ	Compustat	Annual
87	std_dolvol	Volatility of liquidity (dollar trading volume)	Chordia, Subrahmanyam & Anshuman	2001, JFE	CRSP	Monthly
88	std_turn	Volatility of liquidity (share turnover)	Chordia, Subrahmanyam, &Anshuman	2001, JFE	CRSP	Monthly
89	stdacc	Accrual volatility	Bandyopadhyay, Huang & Wirjanto	2010, WP	Compustat	Quarterly
90	stdcf	Cash flow volatility	Huang	2009, JEF	Compustat	Quarterly
91	tang	Debt capacity/firm tangibility	Almeida & Campello	2007, RFS	Compustat	Annual
92	tb	Tax income to book income	Lev & Nissim	2004, TAR	Compustat	Annual
93	turn	Share turnover	Datar, Naik & Radcliffe	1998, JFM	CRSP	Monthly
94	zerotrade	Zero trading days	Liu	2006, JFE	CRSP	Monthly

B.2 Characteristics Importance by each ML Model in Gu et al. (2020)

Figure 10. Characteristics Importance



Note. This figure represents the characteristic's importance in terms of the overall models presented in Gu et al. (2020), which describes the most informative features with dark blue and the least ones with white- reprinted from "Empirical Asset Pricing via Machine Learning" by Gu et al., 2020.

Appendix C

C.1 Data Pre-Processing

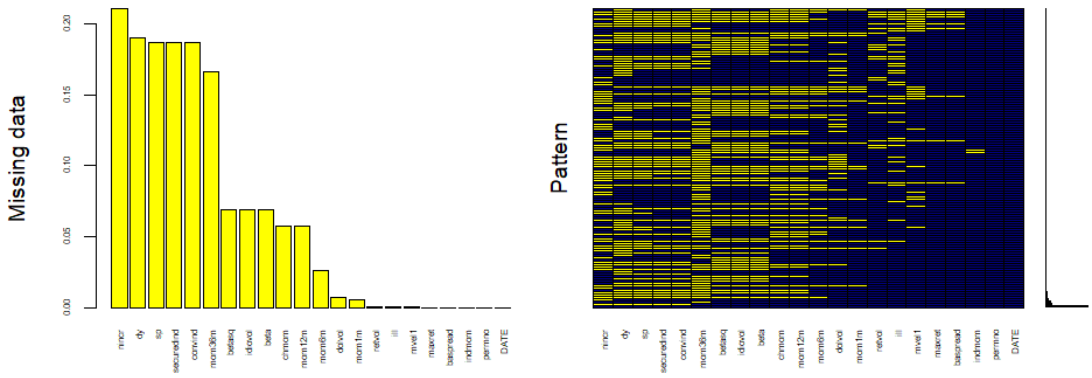
The original data is imported in the following subsection, which contains 869,347 observations for 97 variables. Ninety-four variables correspond to the stock-level characteristics described in Appendix B.1, and two variables represent DATE and Permno, respectively. *Permno* is a unique identifier for one particular stock in the database, and DATE is of the form *yy-mm-dd*. However, as mentioned in section 3.5, this specific study aims to predict individual stocks' return based on the 20-best stock-level characteristics adopted from Gu et al. (2020). Therefore, I extract 20 variables along with DATE and permno shown in the below chunk. The data frame *df* represents these extracted features.

```
#============================
#    Data Pre-processing
#============================

# Set the working directory

setwd("C:\\Users\\adelm\\Desktop\\Masters thesis\\Data")


# Import the data as a data frame

Data<- read.csv("data_20150101.csv")

# Extract the 20 variables needed for prediction

df <- Data[,cbind("permno","DATE","mvel1","mom1m","betasq","dy","maxret",
                  "indmom","retvol","chmom","mom12m","baspread","mom6m",
                  "idiovol","sp","beta","ill","dolvol","mom36m",
                  "securedind","convind","nincr")]
```

Missing Values Imputation

The data frame **df** contains a high number of missing observations. Since the Random Forest algorithm does not support missing values, I replace them with the monthly cross-sectional median for each stock. To this end, I use data manipulation libraries, as mentioned in the below chunk.

```
# Create a function to check for any missing value:

check_missing <- function (x){
  any(is.na(x))
}


# Visualize Missing Values:

library(mice)  # The mice package performs multiple imputations using Fully
               # Conditional Specification (FCS).

library(VIM)   # The VIM package introduces new tools for the visualization
               # of missing values.


mice_plot <- aggr(df, col=c('navyblue','yellow'),
                  numbers=TRUE, sortVars=TRUE,
                  labels=names(df), cex.axis=.5,
                  gap=3, ylab=c("Missing data","Pattern"))
```




```
mice_plot$missings

#For data manipulation, I use the following libraries:

library(tidyverse)

library(dplyr)

library(data.table)

# Replace the missing values of each month by the cross-sectional median
# of all permnos in this month:

old_df <- df %>%
  group_by(DATE) %>%
  mutate_all(funs(ifelse(is.na(.), median(., na.rm = TRUE),.)))
```

C.1.1 Filtration

The filtration procedure includes all the firms presented in each of the 144 months mentioned in section 3.5.1. Thus, resulting in a new data frame containing 412,848 observations for 23 variables represented by **new_df**. After that, each month will include only 2,867 stocks.

```
# Convert the data frame into tibble, which is a modern re-imagining of
# the data.frame:

df <- as_tibble(df)

# Count the number of months for each permno:

count_permno <- old_df %>%
                group_by(permno)%>%
                summarise(n = n())

# Count the number of observations/permnos in each month of the data set:

count_date <- old_df %>%
              group_by(DATE)%>%
              summarise(n = n())

# Extract the permnos found in all the 144 months of the data set:

new_df <- old_df %>%
          group_by(permno)%>%
          filter(n() == 144)

# Count the number of months after extracting the permnos in all the 144 months:

count_permno <- new_df %>%
                group_by(permno)%>%
                summarise(n = n())

# Count the number of observations /permnos in each month for all the 144 months:

count_date <- new_df%>%
              group_by(DATE)%>%
              summarise(n = n())

# This leads to a total of 2,867 observations per month
```

C.1.2 Return Calculation

The dependent variable in the data set, *returns*, is missing. The aim now is to compute the returns by shifting the variable *mom1m* one month into the future, resulting in a new variable *ret*, which is the predicted quantity of interest.

```
# Shift for each permno the variable mom1m one month into the future:

l_df <- new_df %>%
        group_by(permno) %>%
        mutate(ret = lead(mom1m,n=1L,order_by=-DATE,default = NA))

# Relocate the variable "ret" after "mom1m" for comparison:

l_df <- l_df %>%
        relocate(ret, .after = mom1m)

# Arrange the column permno in ascending order:

l_df <- l_df %>%
        arrange(permno)

# Replace the NA of each permno of every first month by
# the median of the same permno of all the years:

l_df <- l_df %>%
       group_by(permno)%>%
       mutate_if(~ any(is.na(.)),~ if_else(is.na(.),
              median(.,na.rm = TRUE),.))

# Finalize the representation of the final data set:

# Arrange the column "DATE":

l_df <- l_df %>%
        arrange(DATE)

# Relocate the column "DATE" to be the first column:

l_df <- l_df %>%
        relocate(DATE, .before = permno)

# Relocate the column dependent variable "ret" before "mvel1":
l_df <- l_df %>%
        relocate(ret, .before = mvel1)

# Change the DATE column from numeric to date format:

l_df <- transform(l_df, DATE = as.Date(as.character(DATE), "%Y%m%d"))
```

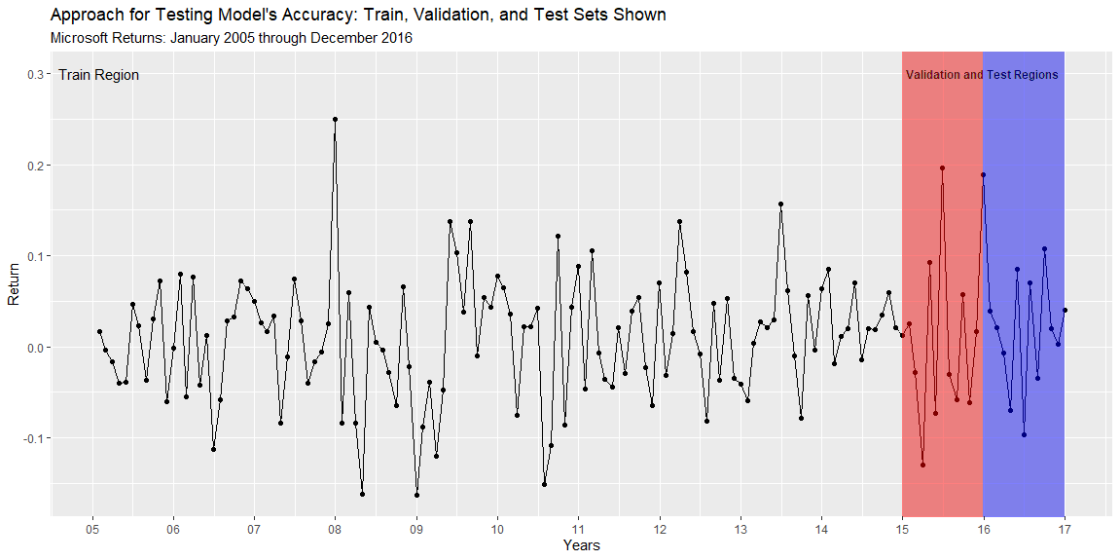
C.1.3 Sample Splitting

Figure 7 in section 3.5.2 provides a detailed explanation of the sample splitting and estimation procedure. The training set starts with 378,444 observations and increases in size as it moves 1-month forward to account for new observations. However, the validation set includes 34,404 observations that remain fixed in size but rolled 1-month into the future. Finally, the testing

set consists of 2,867 observations, which correspond to all the individual stocks each month. Besides, I use Microsoft's returns data to visualize the initial training, validation, and testing sub-samples.

```
#====================================
    # Visualize the Training, Validation, and Test sets
    # of Microsoft for illustration
#====================================

library(ggplot2)  # Create elegant data visualizations using the
                  # grammar of graphics.

 l_df %>%select(DATE, permno, ret)%>%
       filter( as.numeric(permno) == 10107)%>%

   ggplot(aes(x=DATE, y=ret)) +
   geom_line() +
   geom_point() +

   annotate('text',
          x = as.Date('2015-12-31'),
          y = 0.3,
          color = 'black',
          label = 'Validation and Test Regions',
          size = 3.2,
          fontface = 2) +

   geom_rect(xmin = as.numeric(as.Date('2014-12-31')),
           xmax = as.numeric(as.Date('2015-12-31')),
           ymin = -0.3,
           ymax = Inf,
           fill = 'red',
           alpha = 0.002) +

   annotate('text',
          x = as.Date('2016-01-29'),
          y = 0.3,
          color = 'black',
          label = '.',
          size = 0.1) +

   geom_rect(xmin = as.numeric(as.Date('2015-12-29')),
           xmax = as.numeric(as.Date('2016-12-30')),
           ymin = -0.3,
           ymax = Inf,
           fill = 'blue',
           alpha = 0.002) +

   annotate('text',
          x = as.Date('2005-01-31'),
          y = 0.3,
          color = 'black',
          label = 'Train Region',
          fontface =1) +
```

```
  scale_x_date(date_breaks = "1 year", date_labels = "%y") +
  labs(x = "Years", y = "Return",
       title = “Approach for Testing Model’s Accuracy: Train, Validation, and Test Sets Sho
wn”,
       subtitle = "Microsoft Returns: January 2005 through December 2016") +
  theme_gray()
```




```
#=====================================================
#      Training, validation, and testing samples
#      for return prediction of the year 2016
#=====================================================

# January 2016

Train_1 <- l_df %>%
       filter(DATE <= "2015-12-31")

Val_1 <- l_df %>%
       filter(DATE >= "2015-01-30" & DATE <= "2015-12-31")

Test_1 <- l_df %>%
       filter(DATE == "2016-01-29")
#_____________________________________________________________________
___

# February 2016

Train_2 <- l_df %>%
       filter(DATE >= "2005-01-31" & DATE <= "2016-01-29")

Val_2 <- l_df %>%
       filter(DATE >= "2015-02-27" & DATE <= "2016-01-29")

Test_2 <- l_df %>%
```

```
        filter(DATE == "2016-02-29")
#_____________________________________________________________________
___

# March 2016

Train_3 <- l_df %>%
        filter(DATE >= "2005-01-31" & DATE <= "2016-02-29")

Val_3 <- l_df %>%
        filter(DATE >= "2015-03-31" & DATE <= "2016-02-29")

Test_3 <- l_df %>%
        filter(DATE == "2016-03-31")
#_____________________________________________________________________
___

# April 2016

Train_4 <- l_df %>%
        filter(DATE >= "2005-01-31" & DATE <= "2016-03-31")

Val_4 <- l_df %>%
        filter(DATE >= "2015-04-30" & DATE <= "2016-03-31")

Test_4 <- l_df %>%
        filter(DATE == "2016-04-29")
#_____________________________________________________________________
___

# May 2016

Train_5 <- l_df %>%
        filter(DATE >= "2005-01-31" & DATE <= "2016-04-29")

Val_5 <- l_df %>%
        filter(DATE >= "2015-05-29" & DATE <= "2016-04-29")

Test_5 <- l_df %>%
        filter(DATE == "2016-05-31")
#_____________________________________________________________________
___

# June 2016

Train_6 <- l_df %>%
        filter(DATE >= "2005-01-31" & DATE <= "2016-05-31")

Val_6 <- l_df %>%
        filter(DATE >= "2015-06-30" & DATE <= "2016-05-31")

Test_6 <- l_df %>%
        filter(DATE == "2016-06-30")
#_____________________________________________________________________
___

# July 2016
```

```
Train_7 <- l_df %>%
        filter(DATE >= "2005-01-31" & DATE <= "2016-06-30")

Val_7 <- l_df %>%
        filter(DATE >= "2015-07-31" & DATE <= "2016-06-30")

Test_7 <- l_df %>%
        filter(DATE == "2016-07-29")
#_____________________________________________________________________
___

# August 2016

Train_8 <- l_df %>%
        filter(DATE >= "2005-01-31" & DATE <= "2016-07-29")

Val_8 <- l_df %>%
        filter(DATE >= "2015-08-31" & DATE <= "2016-07-29")

Test_8 <- l_df %>%
        filter(DATE == "2016-08-31")
#_____________________________________________________________________
___

# September 2016

Train_9 <- l_df %>%
        filter(DATE >= "2005-01-31" & DATE <= "2016-08-31")

Val_9 <- l_df %>%
        filter(DATE >= "2015-09-30" & DATE <= "2016-08-31")

Test_9 <- l_df %>%
        filter(DATE == "2016-09-30")
#_____________________________________________________________________
___

# October 2016

Train_10 <- l_df %>%
        filter(DATE >= "2005-01-31" & DATE <= "2016-09-30")

Val_10 <- l_df %>%
        filter(DATE >= "2015-10-30" & DATE <= "2016-09-30")

Test_10 <- l_df %>%
        filter(DATE == "2016-10-31")
#_____________________________________________________________________
___

# November 2016

Train_11 <- l_df %>%
        filter(DATE >= "2005-01-31" & DATE <= "2016-10-31")

Val_11 <- l_df %>%
        filter(DATE >= "2015-11-30" & DATE <= "2016-10-31")
```

```
Test_11 <- l_df %>%
        filter(DATE == "2016-11-30")
#_____________________________________________________________________
___

# December 2016

Train_12 <- l_df %>%
  filter(DATE >= "2005-01-31" & DATE <= "2016-11-30")

Val_12 <- l_df %>%
  filter(DATE >= "2015-12-31" & DATE <= "2016-11-30")

Test_12 <- l_df %>%
  filter(DATE == "2016-12-30")
```

C.2 Hyperparameter Tuning via H2O

Two types of grid search that are supported by H2O:

1) Cartesian Grid Search
2) Random Grid Search

I choose to perform a *random grid search* in which I set values for each Hyper-parameter that I want to search over. Hyper_grid1.h2o defines a combination of *ntrees* and *mtries*. Ntrees go from 200 to 500 with an increment of 100, while mtries range from 6 to 15 with an increment of 1.

H2O is an open-source machine learning and predictive analytics platform with core code written in Java. H2O will perform a uniform sampling from the set of all possible hyper-parameter value combinations. When the grid search is complete, the algorithm will use the lowest residual deviance metric to present the model's best results. The results show that a model with mtries = 6 and ntrees = 400 outperforms all other combinations. I use these parameters to model and evaluate the random forest model's performance on the out-of-sample data.

```
#=========================================
#     Installing the H2o package
#=========================================

# The following two commands remove any previously installed H2O
# packages for R:

if ("package:h2o" %in% search()) { detach("package:h2o", unload=TRUE) }
if ("h2o" %in% rownames(installed.packages())) { remove.packages("h2o") }

# Download packages that H2O depends on:

pkgs <- c("RCurl","jsonlite")
for (pkg in pkgs) {
  if (! (pkg %in% rownames(installed.packages()))) { install.packages(pkg) }
}

# Download and install the H2O package for R:

install.packages("h2o", type="source", repos=(c("http://h2o-release.s3.amazonaws.com/h2
o/latest_stable_R")))

# Initialize H2O:

library(h2o)
localH2O = h2o.init(max_mem_size = "16g")

#=========================================
#     Tuning ntrees & mtries via H2O
#=========================================

# Set the predictor names and the response column name:

predictors <- colnames(l_df)[4:23]
response <- "ret"

# Define the hyper-parameter grid:

hyper_grid1.h2o <- list(
  ntrees      = seq(200, 500, by = 100),
  mtries      = seq(6, 15, by = 1))

# Define a search criterion (Random Search):

search_criteria <- list(strategy = "RandomDiscrete")

#Train and validate a random grid of Random Forests:

rf_grid1 <- h2o.grid("randomForest", x = predictors, y = response,
                     grid_id = "rf_grid1",
                     training_frame = as.h2o(Train_1),
                     validation_frame = as.h2o(Val_1),

                     seed = 1,
                     hyper_params = hyper_grid1.h2o,
                     search_criteria = search_criteria)
```

```
Rf_grid_best <- data.frame(mtries,ntrees,model_ids,residual_deviance)
print(Rf_grid_best)

##    mtries ntrees model_ids residual_deviance
## 1       6    400   model_7       0.00411086
## 2       6    500  model_12       0.00411369
## 3       6    200   model_9       0.00411418
## 4       7    200   model_3       0.00413610
## 5       9    200   model_8       0.00420709
## 6      12    500   model_4       0.00431895
## 7      13    300   model_1       0.00436725
## 8      14    500   model_6       0.00441314
## 9      14    300  model_10       0.00441974
## 10     15    200   model_5       0.00444172
## 11     15    400   model_2       0.00444449
## 12     15    300  model_11       0.00444848
```

C.3 Model Evaluation/Prediction

To evaluate the precision of the random forest model predictions for individual stock returns, I use the out-of-sample R-squared as defined in section 3.5.3. The random forest is trained based on the hyper-parameters ntrees = 400, mtries = 6, and max_depth = 6. After that, I evaluate the model based on the Validation set to know if predictions project well on the test data.

```
#==================
#   January 2016
#==================

# Create a function that calculates the out-of-sample R-squared:

Test_r2 <- function(x,y){
  SS.total <- sum((x)^2)
  SS.residual <- sum((x-y)^2)
  r2 <- 1 - (SS.residual/SS.total)
}

# Set the predictor names:

predictors <- colnames(l_df)[-1:-3]

# Set the response column to "ret," which is the return
# of individual stocks:

response <- "ret"

# Build and train the Random Forest model:

library(h2o)
localH2O = h2o.init(max_mem_size = "16g", nthreads = -1)
```

```
rf_model <- h2o.randomForest(x = predictors,
                        y = response,
                        ntrees = 400,
                        max_depth = 6,
                        mtries = 6,
                        nbins_cats = 2867,
                        training_frame = as.h2o(Train_1),
                        validation_frame = as.h2o(Val_1))

# Evaluate the model on the validation set:

perf_1 <- h2o.performance(rf_model, as.h2o(Val_1))

# Calculate the Validation R2:

Val_r2_Jan<- h2o.r2(rf_model, valid = TRUE)

# Predict using the RF model and the testing dataset:

pred_1 <- as.data.frame(h2o.predict(rf_model, as.h2o(Test_1)))

# Combine the predicted values and real observations:

Test_Jan <-pred_1 %>%
        bind_cols(Test_1,pred_1)%>%
        select(DATE,permno, ret, predict...1)%>%
        rename(pred_Jan = predict...1)

# Calculate the out-of-sample R-squared using the function Test_R2:

Test_r2_Jan <-  Test_r2(Test_Jan$ret, Test_Jan$pred_Jan)
```

The same procedure will repeat for 11 months forward. Therefore, to predict the returns for February 2016, the same input parameters will be used; however, I modify the training, validation, and testing subsets such that the estimation procedure follows section 3.5.2.

```
# Create a data frame to combine the results of Validation and test R^2:

Date <- c("Jan 2016", "Feb 2016", "March 2016","April 2016","May 2016",
        "June 2016","July 2016","August 2016","Sep 2016","Oct 2016",
        "Nov 2016","Dec 2016")

Test_R_squared <- c(Test_r2_Jan, Test_r2_Feb, Test_r2_March,
                Test_r2_April, Test_r2_May,Test_r2_June,
                Test_r2_July, Test_r2_August, Test_r2_September,
                Test_r2_October,Test_r2_November, Test_r2_December)

Validation_R_squared <- c(Val_r2_Jan, Val_r2_Feb, Val_r2_March, Val_r2_April,
                    Val_r2_May, Val_r2_June, Val_r2_July,Val_r2_August,
                    Val_r2_September, Val_r2_October, Val_r2_November,
                    Val_r2_December)
```

```
Out_of_sample_R_squared <- c(Test_r2_Jan, Test_r2_Feb, Test_r2_March,
                      Test_r2_April, Test_r2_May,Test_r2_June,
                      Test_r2_July, Test_r2_August, Test_r2_September,
                      Test_r2_October,Test_r2_November,
                      Test_r2_December)

Model <- rep("RandomForest", 12)

Results <- data.frame(Date,Validation_R_squared,Test_R_squared,Model)
Results_output <- data.frame(Date,Out_of_sample_R_squared,Model)
Results
```

```
##          Date Validation_R_squared Test_R_squared       Model
## 1    Jan 2016           0.1854527    0.16981318 RandomForest
## 2    Feb 2016           0.1794951    0.28357498 RandomForest
## 3   March 2016           0.1823445    0.27405374 RandomForest
## 4   April 2016           0.1921007    0.17774508 RandomForest
## 5     May 2016           0.1933183    0.07585074 RandomForest
## 6    June 2016           0.1844245    0.13224659 RandomForest
## 7    July 2016           0.1881956    0.05075565 RandomForest
## 8  August 2016           0.1769710    0.13855476 RandomForest
## 9     Sep 2016           0.1772513    0.29100167 RandomForest
## 10    Oct 2016           0.1780757    0.18993148 RandomForest
## 11    Nov 2016           0.1759774    0.19693195 RandomForest
## 12    Dec 2016           0.1534907    0.06615888 RandomForest
```

```
mean(Out_of_sample_R_squared)
```

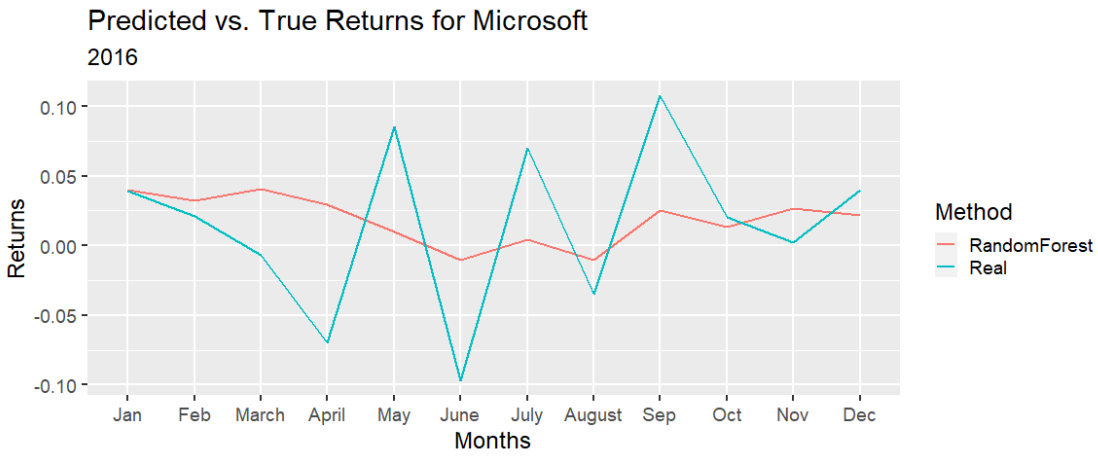
```
## [1] 0.1705516
```

C.4 Return Prediction for Microsoft and Apple Inc.

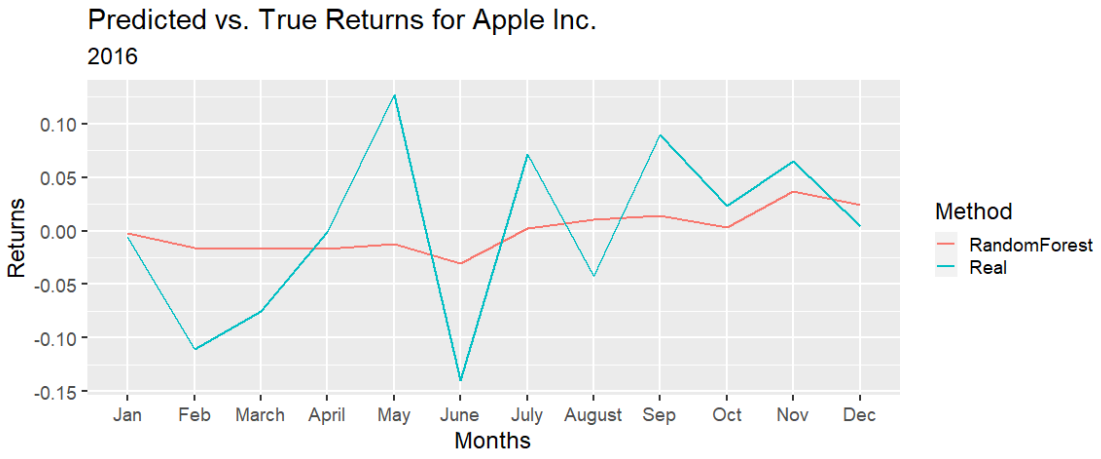
This subsection compares the predicted returns to the real/realized returns of the two largest market capitalization companies in the S&P 500: Microsoft and Apple Inc.

```
library(ggplot2)

# Combine the predictions into one data frame:

DF_combine <- rbindlist(list(Test_Jan,Test_Feb,Test_March,Test_April,
                      Test_May,Test_June,Test_July,Test_August,
                      Test_September, Test_October, Test_November,
                      Test_December), use.names = F)

# Rename the column pred_jan to prediction:

DF_combine <- DF_combine %>%
            rename(prediction = pred_Jan)

# Take the values for Microsoft only:
```

```
per_10107 <- DF_combine %>%
  filter(permno == 10107)

# Extract the predicted returns for Microsoft:

pred_rf_10107 <- data.table(value = per_10107$prediction,
                            Var2 = 1:12, Var1 = "RandomForest")

# Extract the real returns for Microsoft:

pred_true_10107  <- data.table(value = per_10107$ret,
                               Var2 = 1:12, Var1 = "Real")

# Combine the predicted and real returns for Microsoft:

preds_all_10107  <- rbindlist(list(pred_rf_10107 , pred_true_10107 ),
                              use.names = T)

# Plot the results for Microsoft:

ggplot(preds_all_10107, aes(Var2, value, color = as.factor(Var1))) +
  geom_line(alpha = 1, size = 1) +
  labs(x = "Months", y = "Returns",
       title = "Predicted vs. True Returns for Microsoft ",
       subtitle = "2016") +

  guides(color=guide_legend(title="Method")) +
  theme_gray(base_size = 18)+
  scale_x_discrete(limit = c("Jan", "Feb", "March","April","May","June",
                             "July","August","Sep","Oct",
                             "Nov","Dec"))
```



```
# Take the values Apple Inc.:

per_14593 <- DF_combine %>%
          filter(permno == 14593)

# Extract the predicted returns for Apple Inc.:

pred_rf_14593 <- data.table(value = per_14593$prediction,
                      Var2 = 1:12, Var1 = "RandomForest")

# Extract the real returns for Apple Inc.:

pred_true_14593  <- data.table(value = per_14593$ret,
                       Var2 = 1:12, Var1 = "Real")

# Combine the predicted and real returns for Apple Inc.:

preds_all_14593  <- rbindlist(list(pred_rf_14593 , pred_true_14593 ),
                      use.names = T)

# Plot the results for Apple Inc.:

ggplot(preds_all_14593, aes(Var2, value, color = as.factor(Var1))) +
  geom_line(alpha = 1, size = 1) +
  labs(x = "Months", y = "Returns",
      title = "Predicted vs. True Returns for Apple Inc.",
      subtitle = "2016") +

  guides(color=guide_legend(title="Method")) +
  theme_gray(base_size = 18)+
  scale_x_discrete(limit = c("Jan", "Feb", "March","April","May","June",
                       "July","August","Sep","Oct",
                       "Nov","Dec"))
```



Affirmation

I hereby declare that I have composed my Master‘s thesis titled “Empirical Asset Pricing via Random Forest” independently using only those resources mentioned, and that I have as such identified all passages which I have taken from publications verbatim or in substance. I am informed that my thesis might be controlled by anti-plagiarism software.

Neither this thesis, nor any extract of it, has been previously submitted to an examining authority, in this or a similar form.

I have ensured that the written version of this thesis is identical to the version saved on the enclosed storage medium.

Kiel, 11/30/2020

Adel W. Malaeb